Orthogonal Learning Particle Swarm Optimization

Zhi-Hui Zhan, *Student Member, IEEE,* Jun Zhang, *Senior Member, IEEE,* Yun Li, *Member, IEEE,* and Yu-Hui Shi, *Senior Member, IEEE*

***Abstract*—Particle swarm optimization (PSO) relies on its learning strategy to guide its search direction. Traditionally, each particle utilizes its historical best experience and its neighborhood's best experience through linear summation. Such a learning strategy is easy to use, but is inefficient when searching in complex problem spaces. Hence, designing learning strategies that can utilize previous search information (experience) more efficiently has become one of the most salient and active PSO research topics. In this paper, we proposes an orthogonal learning (OL) strategy for PSO to discover more useful information that lies in the above two experiences via orthogonal experimental design. We name this PSO as orthogonal learning particle swarm optimization (OLPSO). The OL strategy can guide particles to fly in better directions by constructing a much promising and efficient exemplar. The OL strategy can be applied to PSO with any topological structure. In this paper, it is applied to both global and local versions of PSO, yielding the OLPSO-G and OLPSO-L algorithms, respectively. This new learning strategy and the new algorithms are tested on a set of 16 benchmark functions, and are compared with other PSO algorithms and some state of the art evolutionary algorithms. The experimental results illustrate the effectiveness and efficiency of the proposed learning strategy and algorithms. The comparisons show that OLPSO significantly improves the performance of PSO, offering faster global convergence, higher solution quality, and stronger robustness.**

***Index Terms*—Global optimization, orthogonal experimental design (OED), orthogonal learning particle swarm optimization (OLPSO), particle swarm optimization (PSO), swarm intelligence.**

I. Introduction

PARTICLE swarm optimization (PSO) algorithm is a global optimization method originally developed by Kennedy and Eberhart [1], [2]. It is a swarm intelligence [3] algorithm that emulates swarm behaviors such as birds flocking and fish schooling. It is a population-based iterative learning algorithm that shares some common characteristics with other evolutionary computation (EC) algorithms [4]. However, PSO searches for an optimum through each particle flying in the search space and adjusting its flying trajectory according to its personal best experience and its neighborhood's best experience rather than through particles undergoing genetic operations like selection, crossover, and mutation [5]. Owing to its simple concept and high efficiency, PSO has become a widely adopted optimization technique and has been successfully applied to many real-world problems [6]–[13].

The salient feature of PSO lies in its learning mechanism that distinguishes the algorithm from other EC techniques [5]. When searching for a global optimum in a hyperspace, particles in a PSO fly in the search space according to guiding rules. It is the guiding rules that make the search effective and efficient. In the traditional PSO, the rules are the mechanism that each particle learns from its own best historical experience and its neighborhood's best historical experience [1]. According to the method of choosing the neighborhood's best historical experience, PSO algorithms are traditionally classified into global version PSO (GPSO) and local version PSO (LPSO). In GPSO, a particle uses the best historical experience of the entire swarm as its neighborhood's best historical experience. In LPSO, a particle uses the best historical experience of the particle in its neighborhood which is defined by some topological structure, such as the ring structure, the pyramid structure, or the von Neumann structure [14]–[16]. Without loss of generality, this paper aims at improving the performance of both the GPSO and the LPSO with the ring structure, where a particle takes its left and right particles (by particle index) as its neighbors [2].

In both GPSO and LPSO, the information of a particle's best experience and its neighborhood's best experience is utilized in a simple way, where the flying is adjusted by a simple learning summation of the two experiences which will be given in (1) in Section II-A. However, this is not necessarily an efficient way to make the best use of the search information in these two experiences. For example, in one case, it may cause an "oscillation" phenomenon [17] because the guidance of the two experiences may be in opposite directions. This is inefficient to the search ability of the algorithm and delays the convergence speed. In another case, the particle may suffer from the "two steps forward, one step back" phenomenon [18] that some components of the solution vector may be improved by one exemplar but may be deteriorated by the other. This is because that one exemplar may have good values on some dimensions of the solution vector while the other exemplar may have good values on some other dimensions. Hence, how to discover

Manuscript received September 16, 2009; revised January 6, 2010 and February 10, 2010; accepted March 6, 2010. Date of publication September 2, 2010; date of current version December 1, 2011. This work was supported in part by the National Natural Science Foundation of China Joint Fund with Guangdong, under Key Project U0835002, by the National High-Technology Research and Development Program ("863" Program) of China, under Grant 2009AA01Z208, and by the Suzhou Science and Technology Project, under Grant SYJG0919.

Z.-H. Zhan and J. Zhang are with the Department of Computer Science, Sun Yat-Sen University, Guangzhou 510275, China, with the Key Laboratory of Digital Life, Ministry of Education, China, and also with the Key Laboratory of Software Technology, Education Department of Guangdong Province, China (e-mail: junzhang@ieee.org).

Y. Li is with the Department of Electronics and Electrical Engineering, University of Glasgow, Glasgow G12 8LT, U.K.

Y.-H. Shi is with the Research and Postgraduate Office and the Department of Electrical and Electronic Engineering, Xi'an Jiaotong-Liverpool University, Jiangsu 215123, China.

Digital Object Identifier 10.1109/TEVC.2010.2052054

more useful information embedded in the two exemplars and thus how to utilize the information to construct an efficient and promising exemplar to guide the particle flying steadily toward the global optimal region are important and challenging research issues that PSO researchers need to pay attention to.

Because orthogonal experimental design (OED) offers an ability to discover the best combination levels for different factors with a reasonably small number of experimental samples [19], [20]. In this paper, we propose to use the OED method to construct a promising learning exemplar. Here, OED is used to discover the best combination of a particle's best historical position and its neighborhood's best historical position. The orthogonal experimental factors are the dimensions of the problem and the levels of each dimension (factor) are the two choices of a particle's best position value and its neighborhood's best position value on this corresponding dimension. This way, the best combination of the two exemplars can be constructed to guide the particle to fly more steadily, rather than oscillatory, because only one constructed exemplar is used for the guidance. It is thus expected for a particle to fly more promisingly toward the global optimum because the constructed exemplar makes the best use of the search information of both the particle's best position and its neighborhood's best position.

In this paper, the OED is used to form an orthogonal learning (OL) strategy for PSO to discover and preserve useful information of a particle's best position and its neighborhood's best position. Owing to the OEDs orthogonal test ability and prediction ability, the OL strategy could construct a guidance exemplar with an ability to predict promising search directions toward the global optimum, therefore results in faster PSO convergence speed and higher solution accuracy. The OL strategy is expected to bring better learning efficiency to PSO and hence better global optimization performance. This learning strategy is applicable to any kind of PSO paradigms, including both GPSO and LPSO. In this paper, the OL strategy-based PSO is termed as orthogonal learning particle swarm optimization (OLPSO). Its advantages will be demonstrated by comparing it with PSOs using traditional learning strategy and with other improved PSOs. Moreover, the OLPSO algorithm is as easy to understand as the traditional PSO and retains the simplicity of PSO.

The rest of the paper is organized as follows. In Section II, the framework of PSO is presented and the researches on improved PSOs are reviewed. In Section III, the OLPSO algorithm is proposed by first discussing the deficiency of traditional learning strategy and then developing the OL strategy. In Section IV, the benchmark functions are used to test OLPSO and to compare it with the PSOs using traditional learning strategy, various improved PSO algorithms, and some state of the art evolutionary algorithms reported in the literatures, in order to verify the effectiveness and efficiency of the OL strategy and the OLPSO in solving global optimization problems. Finally, conclusions are given in Section V.

II. PSO

A. PSO Framework

When searching in a D-dimensional hyperspace, each particle i has a velocity vector $\boldsymbol{V_i} = [v_{i1}, v_{i2}, \ldots, v_{iD}]$ and a position vector $\boldsymbol{X_i} = [x_{i1}, x_{i2}, \ldots, x_{iD}]$ to indicate its current state, where i is a positive integer indexing the particle in the swarm and D is the dimensions of the problem under study. Moreover, particle i will keep its personal historical best position vector $\boldsymbol{P_i} = [p_{i1}, p_{i2}, \ldots, p_{iD}]$. The best position of all the particles in the ith particle's neighborhood (the neighborhood of a particle is defined by a topology structure, e.g., the neighborhood of particle i includes the particles i–1, i, and i+1 in a ring topology structure) is denoted as $\boldsymbol{P_n} = [p_{n1}, p_{n2}, \ldots, p_{nD}]$. The vectors $\boldsymbol{V_i}$ and $\boldsymbol{X_i}$ are initialized randomly and are updated by (1) and (2) generation-by-generation through the guidance of $\boldsymbol{P_i}$ and $\boldsymbol{P_n}$

$$v_{id} = v_{id} + c_1 r_{1d}(p_{id} - x_{id}) + c_2 r_{2d}(p_{nd} - x_{id}) \tag{1}$$

$$x_{id} = x_{id} + v_{id}. \tag{2}$$

Coefficients c_1 and c_2 are acceleration parameters which are commonly set to 2.0 or are adaptively controlled according to the evolutionary states [21]. The r_{1d} and r_{2d} are two randomly generated values within range [0, 1] for the dth dimension. In order to control the flying velocity within a reasonable range, a positive value $V_{\mathrm{MAX}d}$ is used to clamp the updated velocity. If $|v_{id}|$exceeds V_{MAXd}, then it is set to $sign(v_{id})V_{\mathrm{MAX}d}$. However, the updated position x_{id} needs not to be clamped if only the particles within the search space will be evaluated. In this way, all the particles will be drawn back to the range by $\boldsymbol{P_i}$ and $\boldsymbol{P_n}$ which are both within the search space [22].

To control or adjust the flying velocity, however, an inertia weight or a constriction factor is introduced by Shi and Eberhart [23], and Clerc and Kennedy [24], [25], respectively. Using inertia weight ω, (1) is modified to be (3), whilst using the constriction factor χ, (1) is modified to be (4)

$$v_{id} = \omega v_{id} + c_1 r_{1d}(p_{id} - x_{id}) + c_2 r_{2d}(p_{nd} - x_{id}) \tag{3}$$

$$v_{id} = \chi[v_{id} + c_1 r_{1d}(p_{id} - x_{id}) + c_2 r_{2d}(p_{nd} - x_{id})] \tag{4a}$$

where

$$\chi = \frac{2}{\left|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}\right|} \tag{4b}$$

$$\varphi = c_1 + c_2. \tag{4c}$$

In (3), ω usually decreases linearly from 0.9 to 0.4 during the run time [23] whereas χ in (4) is preferably set to be 0.729 together with $c_1 = c_2 = 2.05$ [24], [25]. Moreover, the parameter $V_{\mathrm{MAX}d}$ can be omitted in a PSO with constriction factor. However, as pointed out in [26], the inertia weight and the constriction factor are mathematically equivalent when (4b) and (4c) are met. Without loss of generality, this paper concentrates on the PSO with inertia weight.

B. Improved PSOs

Since its introduction in 1995, PSO has been widely applied to many real-world problems and variants of modified PSOs have been reported in the literature, with enhanced search performance, especially for multimodal optimization

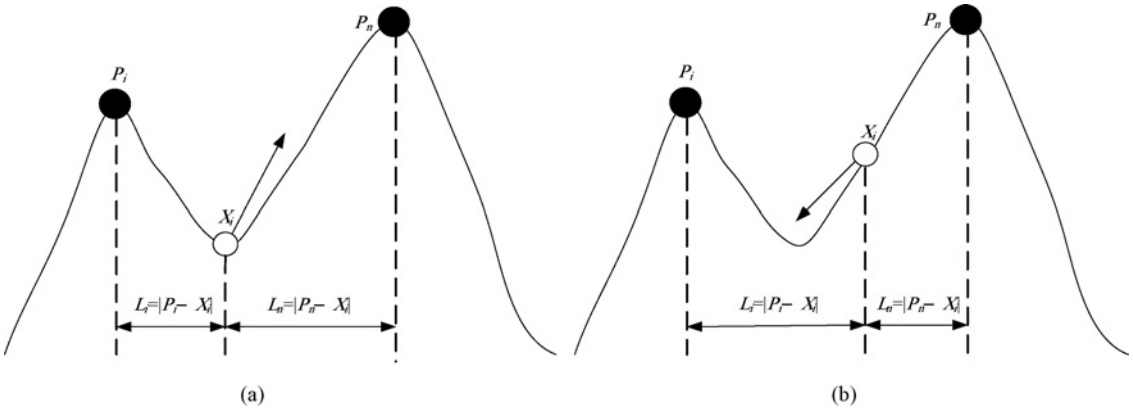

Fig. 1. Oscillation phenomenon in PSO. (a) Flying toward $\boldsymbol{P_n}$ when $L_i < L_n$. (b) Flying toward $\boldsymbol{P_i}$ when $L_i > L_n$.

problems. One active research trend has been to combine PSO with other EC techniques, and this has led to hybrid PSOs. Angeline [27] has first introduced into PSO a selection operator similar to that used in a genetic algorithm (GA). Hybridization of PSO with GA has also been applied to recurrent artificial neural network design [28]. Apart from selection [27], crossover [29], and mutation [30] operations that have been adopted from GAs, more other operations have also been utilized. A PSO algorithm with a cooperative approach called CPSO-S_k was proposed in [18], and a CPSO-H_k algorithm combining the standard PSO with the CPSO-S_k was shown offering a significant improvement over the standard PSO [18]. Further, a re-initialization operation and a self-organizing hierarchical technique have been used together with time-varying acceleration coefficients in order to bring diversity into the population [31]. In addition, Parsopoulos and Vrahatis [17] have proposed a hybrid of deflection and stretching techniques to overcome local minima and have also proposed a repulsion technique to prevent particles moving toward the previously found minima. Inspired by natural evolution, some researchers have introduced niche [32], [33] and speciation techniques [34] into PSO for the purposes of avoiding swarm crowding too closely and of locating as many optimal solutions as possible.

Topological structures of PSO have also been studied widely and various topologies have been suggested. Kennedy [14], [15] has shown that a small neighborhood might work better on complex or multimodal problems whilst a larger neighborhood is better for simpler or unimodal problems. Further, dynamically changing neighborhood structures have been proposed by Suganthan [35], Hu and Eberhart [36], and Liang and Suganthan [37] in order to avoid deficiencies of fixed neighborhoods. In addition, Peram *et al.* [38] have suggested updating the particle velocity by three learning components, two of which are the personal best position and the globally best position, and the other is the particle which has a higher fitness and is nearer to the current particle with a maximal fitness-to-distance ratio. Full information of the entire neighborhood is used to guide the particle in a fully informed particle swarm (FIPS) [39]. That is, all members in the neighborhood can offer their search information fairly or weighted by the distance or fitness of their personal best positions. The cluster centers are calculated in [40] to replace the personal best position or its neighbor's best position, or both. Another modification is the comprehensive learning PSO (CLPSO) [22], which is shown to enhance the diversity of the population by encouraging each particle to learn from different particles on different dimensions, in the metaphor that the best particle which has the highest fitness does not always offer a better value in every dimension.

III. OLPSO

In this section, the traditional PSO learning mechanism is first introduced, followed by the motivations of developing the new OL strategy. Then, the OED method and implementation of the OL strategy are presented. Two complete OLPSO algorithms are derived at the end of the section.

A. Traditional PSO Learning Mechanism

In the traditional PSO, each particle updates its flying velocity and position according to its personal best position and its neighborhood's best position. The concept is simple and appealing, but this learning strategy can cause the "oscillation" phenomenon [17] and the "two steps forward, one step back" phenomenon [18].

The "oscillation" phenomenon is likely to be caused by linear summation of the personal influence and the neighborhood influence. For the clearness and easiness of understanding, we first simplify (3) as (5) by removing the inertia weight component and the random values

$$v_{id} = (p_{id} - x_{id}) + (p_{nd} - x_{id}). \tag{5}$$

In (5), we consider the following case for a maximization problem where the current particle $\boldsymbol{X_i}$ is between its personal best position $\boldsymbol{P_i}$ and its neighborhood's best position $\boldsymbol{P_n}$, as shown in Fig. 1. At first, the distance between $\boldsymbol{P_n}$ and $\boldsymbol{X_i}$ may be farther than the one between $\boldsymbol{P_i}$ and $\boldsymbol{X_i}$, as in Fig. 1(a), then $\boldsymbol{X_i}$ will move toward $\boldsymbol{P_n}$ because of its larger pull. However, as moving toward $\boldsymbol{P_n}$, the distance between $\boldsymbol{P_i}$ and $\boldsymbol{X_i}$ will increase, as shown in Fig. 1(b). In this case, the particle will move toward $\boldsymbol{P_i}$ instead. The oscillation would thus occur and the particle will be puzzled in deciding where to stay [17].

This oscillation phenomenon causes inefficiency to the search ability of the algorithm and delays convergence.

Another related phenomenon of the traditional learning mechanism is the "two step forward, one step back" phenomenon as described in [18]. For example, given a 3-dimension Sphere function $f(X) = x_1^2 + x_2^2 + x_3^2$, whose global minimum point is [0, 0, 0]. Suppose that the current position is $\boldsymbol{X_i}$ = [2, 5, 2], its personal best position is $\boldsymbol{P_i}$ = [0, 2, 5] and its neighborhood's best position is $\boldsymbol{P_n}$ = [5, 0, 1]. The updated velocity is $\boldsymbol{V_i}$ = [1, −8, 2] according to (5), and thus the new position is $\boldsymbol{X_i} = \boldsymbol{X_i} + \boldsymbol{V_i}$ = [3, −3, 4], resulting in a new position with a cost value of 34 which is worse than $\boldsymbol{X_i}$ and $\boldsymbol{P_i}$. Therefore, the particle does not benefit from the learning from $\boldsymbol{P_i}$ and $\boldsymbol{P_n}$ in this generation. However, vectors $\boldsymbol{P_i}$ and $\boldsymbol{P_n}$ indeed possess good information in their structures. For example, if we can discover good dimensions of the two vectors, we can then combine them to form a new guidance vector of $\boldsymbol{P_o}$ = [0, 0, 1] where the first coordinate 0 comes from $\boldsymbol{P_i}$ while the second and the third coordinates 0 and 1 come from $\boldsymbol{P_n}$ (with corresponding dimension). Given the guidance of $\boldsymbol{P_o}$, the updated velocity become $\boldsymbol{V_i} = \boldsymbol{P_o} - \boldsymbol{X_i}$ = [0, 0, 1] − [2, 5, 2] = [−2, −5, −1]; thus the new position is $\boldsymbol{X_i} = \boldsymbol{X_i} + \boldsymbol{V_i}$ = [0, 0, 1], resulting in a new and better position with a cost $f(\boldsymbol{X_i})$ = 1 that makes the particle fly faster toward the global optimum [0, 0, 0].

B. Motivations of the OL Strategy

The first motivation of the OL strategy is that the simple cases above have illustrated the importance of designing a guidance vector $\boldsymbol{P_o}$ that makes an efficient use of search information from $\boldsymbol{P_i}$ and $\boldsymbol{P_n}$. If we exhaustively test all the combinations of $\boldsymbol{P_i}$ and $\boldsymbol{P_n}$ for the best guidance vector $\boldsymbol{P_o}$, 2^D trials are need. This is unrealistic in practice due to the exponential complexity. With the help of OED [19], [20], however, we can devise a relatively good vector from $\boldsymbol{P_i}$ and $\boldsymbol{P_n}$ through only a few experimental tests. In this paper, the OED method is used to test the combinations of $\boldsymbol{P_i}$ and $\boldsymbol{P_n}$ for a better $\boldsymbol{P_o}$. The test factors are the D dimensions and the levels in each factor are 2 for choosing from $\boldsymbol{P_i}$ or $\boldsymbol{P_n}$.

The second motivation of our work is that recent research has shown that incorporating OED in EC algorithms can improve their performance significantly [41]–[50]. The OED method was first introduced into GAs by Zhang and Leung [41] to enhance the crossover operator for multicast routing problems. Leung and Wang [42] proposed another orthogonal GA (OGA/Q) using OED to improve the population initialization and to enhance the crossover operator. Hu *et al.* [43] proposed to use OED on chromosomes to detect surrounding regions for better solutions. Studies in [44] used OED further with a factor analysis that predicts the potentially best combinations. The OED has also been introduced to other optimization algorithms such as simulated annealing [45], [46], ant colony optimization [47], and PSO [48]–[50].

The orthogonal PSO (OPSO) reported in [48] uses an "intelligent move mechanism" (IMM) operation to generate two temporary positions, $\boldsymbol{H}$ and $\boldsymbol{R}$, for each particle $\boldsymbol{X}$, according to the cognitive learning and social learning components, respectively. Then, OED is performed on $\boldsymbol{H}$ and $\boldsymbol{R}$ to obtain the best position $\boldsymbol{X^*}$ for the next move, and then the particle velocity is obtained by calculating the difference between the new position $\boldsymbol{X^*}$ and the current position $\boldsymbol{X}$. Such an IMM was also used in [49] to orthogonally combine the cognitive learning and social learning components to form the next position, and the velocity was determined by the difference between the new position and the current position. The OED in [50] was used to help generate the initial population evenly. Different from previous work and to go steps further, in this paper, we use the OED to form an orthogonal learning strategy, which discovers and preserves useful information in the personal best and the neighborhood best positions in order to construct a promising and efficient exemplar. This exemplar is used to guide the particle to fly toward the global optimal region.

TABLE I
FACTORS AND LEVELS OF THE CHEMICAL EXPERIMENT EXAMPLE

Factors / Levels	A Temp. (°C)	B Time (Min)	C Alkali (%)
1 L_1	80	90	5
2 L_2	85	120	6
3 L_3	90	150	7

The third motivation of developing the OL strategy is that the notion of learning strategy has been very appealing in PSO. A comprehensive learning (CL) strategy has been proposed to extend PSO to CLPSO for improved performance on multimodal functions, but it has been weak in refining solutions, resulting in relatively slow convergence and low-solution accuracy on unimodal functions [22]. Therefore, this paper aims to develop the OL strategy to be able to predict a promising search direction toward the global optimum, for faster convergence and higher solution accuracy. Details of the OED method, the OL strategy, and the OLPSO algorithm will be described in the following sections.

C. OED

In order to illustrate how to use the OED, a simple example is shown in Table I, which arises from chemical experiments. In this example, the aim is to find the best level combination of the three factors involved to increase the conversion ratio. Table I shows that three factors, which will affect experimental results, are the temperature, time and alkali, denoted as factors A, B, and C, respectively. Moreover, there are three levels (different choices) involved in each factor. For example, the temperature can be 80 °C, 85 °C, or 90 °C. Thus, there are in total 3^3 = 27 combinations of experimental designs. However, with the help of OED, one can obtain or predict the best combination by testing only few representative experimental cases.

1) *Orthogonal Array:* The OED method works on a predefined table called an *orthogonal array* (OA). An OA with N factors and Q levels per factor is always denoted by $L_M(Q^N)$, where L denotes the orthogonal array and M is the number of combinations of test cases. For the example shown

TABLE II
DECIDING THE BEST COMBINATION LEVELS OF THE CHEMICAL EXPERIMENTAL FACTORS USING AN OED METHOD

Combinations	A: Temperature °C)	B: Time (Min)	C: Alkali (%)	Results
C_1	(1) 80	(1) 90	(1) 5	$F_1 = 31$
C_2	(1) 80	(2) 120	(2) 6	$F_2 = 54$
C_3	(1) 80	(3) 150	(3) 7	$F_3 = 38$
C_4	(2) 85	(1) 90	(2) 6	$F_4 = 53$
C_5	(2) 85	(2) 120	(3) 7	$F_5 = 49$
C_6	(2) 85	(3) 150	(1) 5	$F_6 = 42$
C_7	(3) 90	(1) 90	(3) 7	$F_7 = 57$
C_8	(3) 90	(2) 120	(1) 5	$F_8 = 62$
C_9	(3) 90	(3) 150	(2) 6	$F_9 = 64$
Levels	Factor Analysis			
L_1	$(F_1 + F_2 + F_3)/3 = 41$	$(F_1 + F_4 + F_7)/3 = 47$	$(F_1 + F_6 + F_8)/3 = 45$	
L_2	$(F_4 + F_5 + F_6)/3 = 48$	$(F_2 + F_5 + F_8)/3 = \mathbf{55}$	$(F_2 + F_4 + F_9)/3 = \mathbf{57}$	
L_3	$(F_7 + F_8 + F_9)/3 = \mathbf{61}$	$(F_3 + F_6 + F_9)/3 = 48$	$(F_3 + F_5 + F_7)/3 = 48$	
OED Results	A3	B2	C2	

in Table I, the $L_9(3^4)$ OA given by (6) is suitable [20]

$$L_9(3^4) = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 2 & 2 \\ 1 & 3 & 3 & 3 \\ 2 & 1 & 2 & 3 \\ 2 & 2 & 3 & 1 \\ 2 & 3 & 1 & 2 \\ 3 & 1 & 3 & 2 \\ 3 & 2 & 1 & 3 \\ 3 & 3 & 2 & 1 \end{bmatrix}. \quad (6)$$

The OA in (6) has four columns, meaning that it is suitable for the problems with at most four factors. As any sub columns of an OA is also an OA, we can to use only the first three columns (or arbitrary three columns) of the array for the experiment. For example, the first three columns in the first row is [1, 1, 1], meaning that in this experiment, the first factor (temperature), the second factor (time), and the third factor (alkali) are all designed to the first level, that is, 80 °C, 90 minutes, and 5% as given in Table I. Similarly, combination of [1, 2, 2] is used in the second experiment, and so on. The total of nine experiments specified by the $L_9(3^4)$ are presented in Table II.

2) *Factor Analysis:* The ability of discovering the best combination of levels is through the *factor analysis* (FA). The FA is based on the experimental results of all the M cases of the OA. The FA results are shown in Table II and the process is described as follows.

Let f_m denote the experimental result of the mth ($1 \le m \le M$) combination and S_{nq} denote the effect of the qth ($1 \le q \le Q$) level in the nth ($1 \le n \le N$) factor. The calculation of S_{nq} is to add up all the f_m in which the level is q in the nth factor, and then divide the total count of z_{mnq}, as shown in (7) where z_{mnq} is 1 if the mth experimental test is with the qth level of the nth factor, otherwise, z_{mnq} is 0

$$S_{nq} = \frac{\sum_{m=1}^{M} f_m \times z_{mnq}}{\sum_{m=1}^{M} z_{mnq}}. \quad (7)$$

In this way, the effect of each level on each factor can be calculated and compared, as shown in Table II. For example, when we calculate the effect of level 1 on factor A, denoted by element A1, the experimental results of C_1, C_2, and C_3 are summed up for (7) because only these combinations are involved in level 1 of factor A. Then, the sum divides the combination number (3 in this case) to yield S_{nq} (S_{A1} in this case). With all the S_{nq} calculated, the best combination of the levels can be determined by selecting the level of each factor that provides the highest-quality S_{nq}. For a maximization problem, the larger the S_{nq} is, the better the qth level on factor n will be. Otherwise, vice versa. As in the maximization example shown in Table II, the best result is the combination of A3, B2, and C2. Although the combination of (A3, B2, C2) itself does not exist in the nine combinations tested, it is discovered by the FA process.

D. OL Strategy

Using the OED method, the original PSO can be modified as an OLPSO with an OL strategy that combines information of $\boldsymbol{P_i}$ and $\boldsymbol{P_n}$ to form a better guidance vector $\boldsymbol{P_o}$. The particle's flying velocity is thus changed as

$$v_{id} = \omega v_{id} + c r_d (p_{od} - x_{id}) \quad (8)$$

where ω is the same as in (3) and c is fixed to be 2.0, the same as c_1 and c_2, and r_d is a random value uniformly generated within the interval [0, 1].

The guidance vector $\boldsymbol{P_o}$ is constructed for each particle i, respectively, from $\boldsymbol{P_i}$ and $\boldsymbol{P_n}$ as

$$\boldsymbol{P_o} = \boldsymbol{P_i} \oplus \boldsymbol{P_n} \quad (9)$$

where the symbol $\oplus$ stands for the OED operation. Therefore, the value p_{od} comes from p_{id} or p_{nd} as the construct result of OED. With this efficient learning exemplar $\boldsymbol{P_o}$, particle i adjusts its flying velocity, position and updates its personal best position in every generation. In order to avoid the guidance changing the direction frequently, the vector $\boldsymbol{P_o}$ will be used as the exemplar for a certain number of generations until it cannot lead the particle to a better position any more. For example, if the personal best position $\boldsymbol{P_i}$ has not been improved for G generations, then particle i will reconstruct a new $\boldsymbol{P_o}$ by using $\boldsymbol{P_i}$ and $\boldsymbol{P_n}$. On the other hand, as $\boldsymbol{P_o}$ is used for some time until it cannot improve the position, one problem should be addressed is how to use the information that comes from $\boldsymbol{P_i}$ and $\boldsymbol{P_n}$ immediately after $\boldsymbol{P_i}$ and $\boldsymbol{P_n}$ go to a better position during the search process. In our implementations, vector $\boldsymbol{P_o}$ stores only the index of $\boldsymbol{P_i}$ and $\boldsymbol{P_n}$, not the copy of the real position values. That is, p_{od} only indicates that the dth dimension is guided by $\boldsymbol{P_i}$ or $\boldsymbol{P_n}$, it does not store the current value of p_{id} or p_{nd}. Thus, in the OLPSO algorithm, when $\boldsymbol{P_i}$ or $\boldsymbol{P_n}$ moves to a better position, the new information will be used immediately by the particle through $\boldsymbol{P_o}$.

The construction process of $\boldsymbol{P_o}$ is described as the following six steps.

Step 1) An OA is generated as $L_M(2^D)$ where $M = 2^{\lceil \log_2(D+1) \rceil}$, using the procedure as given in Appendix.

Step 2) Make up M tested solutions $\boldsymbol{X}_j$ ($1 \le j \le M$) by selecting the corresponding value from $\boldsymbol{P_i}$ or $\boldsymbol{P_n}$ according to the OA. Here, if the level value in the

OA is 1, then the corresponding factor (dimension) selects $\boldsymbol{P_i}$; otherwise, selects $\boldsymbol{P_n}$.

Step 3) Evaluate each tested solution $\boldsymbol{X_j}$ ($1 \le j \le M$), and record the best (with best fitness) solution $\boldsymbol{X_b}$.

Step 4) Calculate the effect of each level on each factor and determine the best level for each factor using (7).

Step 5) Derive a predictive solution $\boldsymbol{X_p}$ with the levels determined in Step 4 and evaluate $\boldsymbol{X_p}$.

Step 6) Compare $f(\boldsymbol{X_b})$ and $f(\boldsymbol{X_p})$ and the level combination of the better solution is used to construct the vector $\boldsymbol{P_o}$.

In the above process, each of the D dimensions is regarded as a factor and therefore there are D factors in the OED. This results in $M = 2^{\lceil \log_2(D+1) \rceil}$ orthogonal combinations because the level of each factor is two [20]. Therefore the M is no larger than $2D$, which is significantly smaller than the total number of combinations 2^D. A method to further reduce the number of the orthogonal combinations is to divide the dimensions into several disjoint groups and regard each group as a factor. This method also may be good for the problems whose dimensions are not independent of each other. Unfortunately, how many groups should be divided and how to assign different dimensions to different groups are usually problem-dependent and difficult to decide [44]. Therefore, since the problem characteristic is usually unknown, it may be a good choice to regard each dimension as a factor. In fact, by using OED, some factors can be regarded as in the same group when they are with the same level. Therefore, it is without loss of generality to regard each dimension as a factor.

E. OLPSO

The OL strategy is a generic operator and can be applied to any kind of topology structure. If the OL is used for the GPSO, then $\boldsymbol{P_n}$ is $\boldsymbol{P_g}$. If it is used for the LPSO, then $\boldsymbol{P_n}$ is $\boldsymbol{P_l}$. Either for a global or a local version, when constructing the vector of $\boldsymbol{P_o}$, if $\boldsymbol{P_i}$ is the same as $\boldsymbol{P_n}$ (e.g., for the globally best particle, $\boldsymbol{P_i}$ and $\boldsymbol{P_g}$ are identical vectors), the OED makes no contribution. In such a case, OLPSO will randomly select another particle $\boldsymbol{P_r}$, and then construct $\boldsymbol{P_o}$ by using the information of $\boldsymbol{P_i}$ and $\boldsymbol{P_r}$ through the OED.

The flowchart of OLPSO is shown in Fig. 2. As discussed in the previous section, the particle will use vector $\boldsymbol{P_o}$ as the learning exemplar steadily and reconstruct the $\boldsymbol{P_o}$ only after a stagnation of $\boldsymbol{P_i}$ for G generations. As can be imagined, if G is too small, the particles will reconstruct the guidance exemplar $\boldsymbol{P_o}$ frequently. This may waste computations on OED when it is not indeed necessary. Also, the search direction will not be steady if $\boldsymbol{P_o}$ changes frequently. On the other hand, if G is too large, the particles will waste much computation on the local optima with a $\boldsymbol{P_o}$ which is not effective any longer.

In order to investigate the influence of G on the performance of the OLPSO algorithm, empirical studies are carried out on relevant functions, namely the Sphere, Rosenbrock, Schwefel, Rastrigrin, Ackley, and Grienwank functions listed in Table III as the f_1, f_3, f_5, f_6, f_7, and f_8, respectively. Different values for G from 0 to 10 are tested, and two OLPSO versions that based on a global topology (OLPSO-G) and a local topology

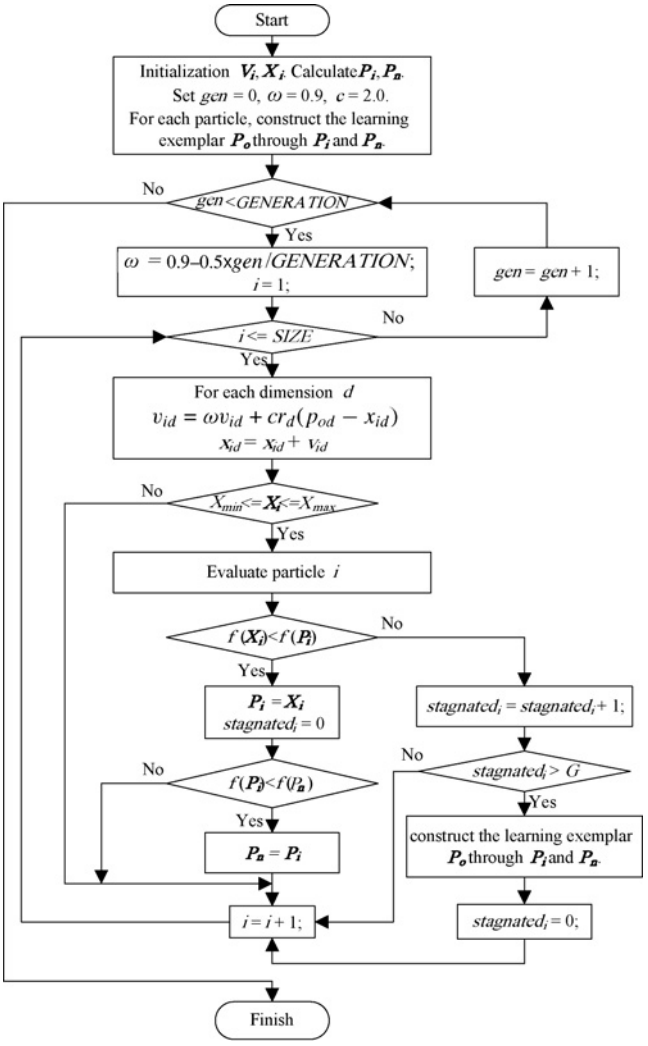

Fig. 2. Flowchart of an OLPSO algorithm.

(OLPSO-L) are simulated. The results of the investigation are shown in Fig. 3(a) and (b) with averagely 25 independent runs for the OLPSO-G and the OLPSO-L, respectively. The figures reveal that a value of G around 5 offers the best performance. This also indicates OLPSO indeed benefits from the OL strategy by the steadily guidance of a promising learning exemplar. Therefore, a reconstruction gap of $G = 5$ is used in this paper.

IV. Experimental Verification and Comparisons

A. Functions Tested and PSOs Compared

Sixteen benchmark functions listed in Table III are used in the experimental tests. These benchmark functions are widely adopted in benchmarking global optimization algorithms [22], [51], [56]. In this paper, the functions are divided into three groups. The first group includes four unimodal functions, where f_1 and f_2 are simple unimodal, f_3 (Rosenbrock) is unimodal in a 2-D or 3-D search space but can be treated as a multimodal function in high-dimensional cases [52]–[54], and f_4 is with noisy perturbation. The second group includes six complex multimodal functions with high dimensionality. The

TABLE III
SIXTEEN TEST FUNCTIONS USED IN THE COMPARISON

	Test Function	D	Search Range	Initialization Range	Global Opt. x^*	$f_{\min}$	Accept	Name
Unimodal	$f_1(x) = \sum_{i=1}^{D} x_i^2$	30	$[-100, 100]^D$	$[-100, 50]^D$	$\{0\}^D$	0	1×10^{-6}	Sphere [51]
	$f_2(x) = \sum_{i=1}^{D} \lvert x_i \rvert + \prod_{i=1}^{D} \lvert x_i \rvert$	30	$[-10, 10]^D$	$[-10, 5]^D$	$\{0\}^D$	0	1×10^{-6}	Schwefel'sP2.22[51]
	$f_3^*(x) = \sum_{i=1}^{D-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	30	$[-10, 10]^D$	$[-10, 10]^D$	$\{1\}^D$	0	100	Rosenbrock [51]†
	$f_4(x) = \sum_{i=1}^{D} ix_i^4 + random[0, 1)$	30	$[-1.28, 1.28]^D$	$[-1.28, 0.64]^D$	$\{0\}^D$	0	0.01	Noise [51]
Multimodal	$f_5(x) = 418.9829 \times D - \sum_{i=1}^{D} x_i \sin(\sqrt{\lvert x_i \rvert})$	30	$[-500, 500]^D$	$[-500, 500]^D$	$\{420.96\}^D$	0	2000	Schwefel [51]
	$f_6(x) = \sum_{i=1}^{D} [x_i^2 - 10\cos(2\pi x_i) + 10]$	30	$[-5.12, 5.12]^D$	$[-5.12, 2]^D$	$\{0\}^D$	0	100	Rastrigin [51]
	$f_7(x) = -20\exp(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}) - \exp(\frac{1}{D}\sum_{i=1}^{D}\cos 2\pi x_i) + 20 + e$	30	$[-32, 32]^D$	$[-32, 16]^D$	$\{0\}^D$	0	1×10^{-6}	Ackley [51]
	$f_8(x) = 1/4000 \sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D} \cos(x_i/\sqrt{i}) + 1$	30	$[-600, 600]^D$	$[-600, 200]^D$	$\{0\}^D$	0	1×10^{-6}	Griewank [51]†
	$f_9(x) = \frac{\pi}{D}\{10\sin^2(\pi y_1) + \sum_{i=1}^{D-1}(y_i - 1)^2[1 + 10\sin^2(\pi y_{i+1})] + (y_D - 1)^2\} + \sum_{i=1}^{D} u(x_i, 10, 100, 4)$ where $y_i = 1 + \frac{1}{4}(x_i + 1)$, $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m, & x_i > a \\ 0, & -a \le x_i \le a \\ k(-x_i - a)^m, & x_i < -a \end{cases}$	30	$[-50, 50]^D$	$[-50, 25]^D$	$\{0\}^D$	0	1×10^{-6}	Generalized Penalized [51]
	$f_{10}(x) = \frac{1}{10}\{\sin^2(3\pi x_1) + \sum_{i=1}^{D-1}(x_i - 1)^2[1 + \sin^2(3\pi x_{i+1})] + (x_D - 1)^2[1 + \sin^2(2\pi x_D)]\} + \sum_{i=1}^{D} u(x_i, 5, 100, 4)$	30	$[-50, 50]^D$	$[-50, 25]^D$	$\{0\}^D$	0	1×10^{-6}	
Rotated and Shifted	$f_{11}(y) = 418.9828 \times D - \sum_{i=1}^{D} z_i$, where $z_i = \begin{cases} y_i \sin(\sqrt{\lvert y_i \rvert}), & \text{if } \lvert y_i \rvert \le 500 \\ 0, & \text{otherwise} \end{cases}$, $y_i = y_i' + 420.96$, where $y' = M^*(x - 420.96)$, M is an orthogonal matrix	30	$[-500, 500]^D$	$[-500, 500]^D$	$\{420.96\}^D$	0	5000	Rotated Schwefel [22]†
	$f_{12}(y) = \sum_{i=1}^{D} [y_i^2 - 10\cos(2\pi y_i) + 10]$ where $y = M^*x$, M is an orthogonal matrix	30	$[-5.12, 5.12]^D$	$[-5.12, 2]^D$	$\{0\}^D$	0	100	Rotated Rastrigin [22]†
	$f_{13}(y) = -20\exp(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} y_i^2}) - \exp(\frac{1}{D}\sum_{i=1}^{D}\cos 2\pi y_i) + 20 + e$ where $y = M^*x$, M is an orthogonal matrix	30	$[-32, 32]^D$	$[-32, 16]^D$	$\{0\}^D$	0	1×10^{-6}	Rotated Ackley [22]†
	$f_{14}(y) = 1/4000 \sum_{i=1}^{D} y_i^2 - \prod_{i=1}^{D} \cos(y_i/\sqrt{i}) + 1$ where $y = M^*x$, M is an orthogonal matrix	30	$[-600, 600]^D$	$[-600, 200]^D$	$\{0\}^D$	0	1×10^{-6}	Rotated Griewank [22]†
	$f_{15}(x) = \sum_{i=1}^{D-1}(100(z_i^2 - z_{i+1})^2 + (z_i - 1)^2) + f\_bias_6$, $z = x - o + 1$, $o = [o_1, o_2, \cdots, o_D]$: the shifted global optimum	30	$[-100, 100]^D$	$[-100, 100]^D$	$\boldsymbol{o}$	390	490	Shifted Rosenbrock [56]†
	$f_{16}(x) = \sum_{i=1}^{D}(z_i^2 - 10\cos(2\pi z_i) + 10) + f\_bias_9$, $z = x - o$, $o = [o_1, o_2, \cdots, o_D]$: the shifted global optimum	30	$[-5, 5]^D$	$[-100, 100]^D$	$\boldsymbol{o}$	−330	−230	Shifted Rastrigin [56]

f_3^* is unimodal in a 2-D or 3-D search space but can be treated as a multimodal function in high-dimensional cases.
† The function is non-separable.

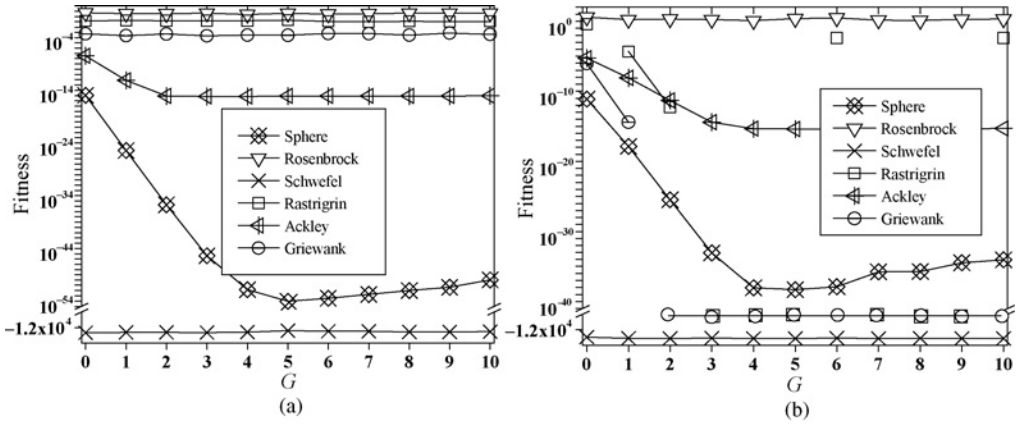

Fig. 3. OLPSO performance with various values of the reconstruction gap G. (a) OLPSO with a global topology. (b) OLPSO with a local topology.

TABLE IV
PSO ALGORITHMS FOR COMPARISON

Algorithm	Parameters Settings	Reference
GPSO	ω: 0.9∼0.4, $c_1 = c_2 = 2.0$, $V_{MAXd} = 0.2 \times$ Range	[23]
LPSO	ω: 0.9∼0.4, $c_1 = c_2 = 2.0$, $V_{MAXd} = 0.2 \times$ Range	[15]
SPSO	$\omega = 0.721$, $c_1 = c_2 = 1.193$, $K = 3$, without V_{MAX}	[55]
FIPS	$\chi = 0.729$, $\sum c_i = 4.1$, $V_{MAXd} = 0.5 \times$ Range	[39]
HPSO-TVAC	ω: 0.9∼0.4, c_1: 2.5∼0.5, c_2: 0.5∼2.5, $V_{MAXd} = 0.5 \times$ Range	[31]
DMS-PSO	ω: 0.9∼0.2, $c_1 = c_2 = 2.0$, $m = 3$, $R = 5$, $V_{MAXd} = 0.2 \times$ Range	[37]
CLPSO	ω: 0.9∼0.4, $c = 1.49445$, $m = 7$, $V_{MAXd} = 0.2 \times$ Range	[22]
OPSO	ω: 0.9∼0.4, $c_1 = c_2 = 2.0$, $V_{MAXd} = 0.5 \times$ Range	[48]
OLPSO	ω: 0.9∼0.4, $c = 2.0$, $G = 5$, $V_{MAXd} = 0.2 \times$ Range	–

last group includes four rotated multimodal functions and 2 shifted functions defined in [56].

Table III gives the global optimal solution (column 5) and the global optimal value $f_{\min}$ (column 6). Moreover, biased initializations (column 4) are used according to the definitions in [22] for the functions whose global solution point is at the center of the search range. "Accept" (column 7) is also defined for each test function. If a solution found by an algorithm falls between the acceptable value and the actual global optimum $f_{\min}$ (column 5), the run is judged to be successful.

Variant PSO algorithms, as detailed in Table IV, are used for comparisons. The parameter configurations are all based on the suggestions in the corresponding references. The first two are traditional PSOs of GPSO [23] and LPSO [15]. The third is the Standard PSO (SPSO) [55]. SPSO is the current standard which is improved by using a random topology, refer to http://www.particleswarm.info/Programs.html#Standard_PSO_2007 for more details. The fourth is a "fully informed" PSO (FIPS) [39] that uses all the neighbors to influence the flying velocity. The fifth is a "performance-improvement" PSO by improving the acceleration coefficients, namely hierarchical PSO with time-varying acceleration coefficients (HPSO-TVAC) [31]. The sixth is a dynamic multi-swarm PSO (DMS-PSO) [37] which is designed to improve the topological structure in a dynamic way. The seventh, CLPSO [22], aims to offer a better performance for multimodal functions by using a CL strategy. The eighth, the OPSO [48] algorithm, aims to improve the algorithm by using an OED to generate a better position, not by constructing a learning exemplar as proposed in this paper. These PSO variants are used for comparisons because they are typical PSOs that are reported to perform well on their studied problems. Moreover, they span a wide time interval from 1998 to 2008, which witness the developments of PSO on variant aspects. For OLPSO developed in this paper, we implement the OL strategy in both the global and the local version PSO, resulting in two OLPSO algorithms, the OLPSO-G and the OLPSO-L, respectively. Both will be compared with GPSO, LPSO, SPSO, FIPS, HPSO-TVAC, DMS-PSO, CLPSO, and OPSO.

For a fair comparison among all the PSOs, they are tested using the same population size of 40. As the SPSO in [55] automatically computes a size of 20 for 30-D functions, we both use SPSO with 20 and 40 particles in the experiments. The two SPSO variants are denoted as SPSO-20 and SPSO-40, respectively. Furthermore, all the algorithms use the same maximum number of function evaluations (FEs) 2×10^5 in each run for each test function. Note that the number of FEs consumed during the construction of the guidance exemplar $\boldsymbol{P_o}$ in OLPSO are included in this maximum FEs number allowed. Also notice that an $L_{32}(2^{31})$ OA [20], [44] is suitable for all the test functions because they are all 30 dimensional. For the purpose of reducing statistical errors, each algorithm is tested 25 times independently for every function and the mean results are used in the comparison.

B. Solution Accuracy with OL Strategy

The solutions obtained by OLPSOs are compared with the ones obtained by PSOs without OL strategy in Table V. Table V compares the mean values and the standard deviations of the solutions found. The best results are marked in boldface. The *t*-test results between OLPSO-G and GPSO, and OLPSO-L and LPSO are also given, respectively.

1) *Unimodal Functions:* For the four unimodal functions, the results show that OLPSOs generally outperform the traditional PSOs. For example, OLPSO-G does better than GPSO on functions f_1, f_2, and f_3 whilst OLPSO-L outperforms LPSO on functions f_1, f_2, f_3, and f_4. The experimental results show that the OL strategy brings solution with much higher accuracy to the problem. For the very simple unimodal functions f_1 and f_2, OLPSO-G provides solutions with the highest quality. However, as the problem becomes more complex, even become multimodal in high dimension, such as the Rosenbrock's function (f_3), the performance of OLPSO-L is much better. This is in coincidence with the general observation that a LPSO does better than a GPSO on complex problems. This is because a LPSO draws experience from locally best particles, as opposed to the interim global best, and hence avoids a premature convergence, although it could converge more slowly. As for the Noise function (f_4), we can observe that OLPSO-G does not show an advantage. This is perhaps because the effect of the OL strategy is largely canceled out by the random fluctuation.

The plots in Fig. 4 show the convergence progress of the mean solution values of the 25 trials during the run for functions f_1 and f_3. It is apparent that OLPSOs perform better than the traditional PSOs in terms of final solution and convergence speed. It can be observed from the figures that OLPSOs with the OL strategy converge considerably faster than the traditional PSOs (GPSO and LPSO) without an OL strategy.

2) *Multimodal Functions:* As the efficiency of the OL strategy provides PSO an ability to discover, preserve, and utilize useful information of the learning exemplars, it is expected that OLPSO can avoid local optima and bring about improved performance on multimodal functions. Indeed, the experimental results for functions f_5–f_{10} given in Table V support this intuition. OLPSO-G surpasses GPSO on all the six multimodal functions. OLPSO-L yields the best performance among the four PSOs on all the six multimodal functions, in terms of mean solutions and standard deviations. In comparison, GPSO can only reach the global optimum on function f_7 and f_9 while LPSO on functions f_7, f_9, and f_{10}. Best of all, OLPSO-L is able to find the global optimum on all the functions and only OLPSO-L can show significantly improved performance in reaching the global

TABLE V
SOLUTIONS ACCURACY (MEAN AND STANDARD DEVIATION) COMPARISONS BETWEEN PSOS WITH AND WITHOUT THE OL STRATEGY

Func	GPSO	OLPSO-G	t-Test	LPSO	OLPSO-L	t-Test
f_1	$\mathbf{2.05 \times 10^{-32} \pm 3.56 \times 10^{-32}}$	$\mathbf{4.12\times10^{-54}\pm6.34\times10^{-54}}$	2.88†	$3.34\times10^{-14}\pm5.39\times10^{-14}$	$\mathbf{1.11\times10^{-38}\pm1.28\times10^{-38}}$	3.10‡
f_2	$1.49\times10^{-21}\pm3.60\times10^{-21}$	$\mathbf{9.85\times10^{-30}\pm1.01\times10^{-29}}$	2.07†	$1.70\times10^{-10}\pm1.39\times10^{-10}$	$\mathbf{7.67\times10^{-22}\pm5.63\times10^{-22}}$	6.12‡
f_3	40.70 ± 32.19	$\mathbf{21.52\pm29.92}$	2.18†	28.08 ± 21.79	$\mathbf{1.26\pm1.40}$	6.14‡
f_4	$\mathbf{9.32 \times 10^{-3} \pm 2.39 \times 10^{-3}}$	$1.16\times10^{-2}\pm4.10\times10^{-3}$	−2.38†	$2.28\times10^{-2}\pm5.60\times10^{-3}$	$\mathbf{1.64\times10^{-2}\pm3.25\times10^{-3}}$	4.96‡
f_5	$2.48\times10^{3}\pm2.97\times10^{2}$	$\mathbf{3.84\times10^{2}\pm2.17\times10^{2}}$	28.53†	$3.16\times10^{3}\pm4.06\times10^{2}$	$\mathbf{3.82\times10^{-4}\pm0}$	38.95‡
f_6	26.03 ± 7.27	$\mathbf{1.07\pm0.99}$	17.00†	35.07 ± 6.89	$\mathbf{0\pm0}$	25.46‡
f_7	$1.31\times10^{-14}\pm2.08\times10^{-15}$	$\mathbf{7.98\times10^{-15}\pm2.03\times10^{-15}}$	8.80†	$8.20\times10^{-08}\pm6.73\times10^{-08}$	$\mathbf{4.14\times10^{-15}\pm0}$	6.09‡
f_8	$2.12\times10^{-2}\pm2.18\times10^{-2}$	$\mathbf{4.83\times10^{-3}\pm8.63\times10^{-3}}$	3.50†	$1.53\times10^{-3}\pm4.32\times10^{-3}$	$\mathbf{0\pm0}$	1.77
f_9	$2.23\times10^{-31}\pm7.07\times10^{-31}$	$\mathbf{1.59\times10^{-32}\pm1.03\times10^{-33}}$	1.46	$8.10\times10^{-16}\pm1.07\times10^{-15}$	$\mathbf{1.57\times10^{-32}\pm2.79\times10^{-48}}$	3.80‡
f_{10}	$1.32\times10^{-3}\pm3.64\times10^{-3}$	$\mathbf{4.39\times10^{-4}\pm2.20\times10^{-3}}$	1.03	$3.26\times10^{-13}\pm3.70\times10^{-13}$	$\mathbf{1.35\times10^{-32}\pm5.59\times10^{-48}}$	4.41‡
f_{11}	$4.61\times10^{3}\pm6.21\times10^{2}$	$\mathbf{4.00\times10^{3}\pm6.08\times10^{2}}$	3.51†	$4.50\times10^{3}\pm3.97\times10^{2}$	$\mathbf{3.13\times10^{3}\pm1.24\times10^{3}}$	5.28‡
f_{12}	60.02 ± 15.98	$\mathbf{46.09\pm12.88}$	3.39†	53.36 ± 13.99	$\mathbf{53.35\pm13.35}$	0.00
f_{13}	1.93 ± 0.96	$\mathbf{7.69\times10^{-15}\pm1.78\times10^{-15}}$	10.01†	1.55 ± 0.45	$\mathbf{4.28\times10^{-15}\pm7.11\times10^{-16}}$	17.44‡
f_{14}	$1.80\times10^{-2}\pm2.41\times10^{-2}$	$\mathbf{1.68\times10^{-3}\pm4.13\times10^{-3}}$	3.33†	$1.68\times10^{-3}\pm3.47\times10^{-3}$	$\mathbf{4.19\times10^{-8}\pm2.06\times10^{-7}}$	2.42‡
f_{15}	427.93 ± 54.98	$\mathbf{424.75\pm34.80}$	0.24	432.33 ± 43.41	$\mathbf{415.95\pm23.96}$	1.65
f_{16}	-223.18 ± 38.58	$\mathbf{-328.57 \pm 1.04}$	13.65†	-234.95 ± 18.82	$\mathbf{-330 \pm 1.64 \times 10^{-14}}$	25.36‡

†The value of t with 48 degrees of freedom is significant at $\alpha = 0.05$ by a two-tailed test between GPSO and OLPSO-G.

‡The value of t with 48 degrees of freedom is significant at $\alpha = 0.05$ by a two-tailed test between LPSO and OLPSO-L.

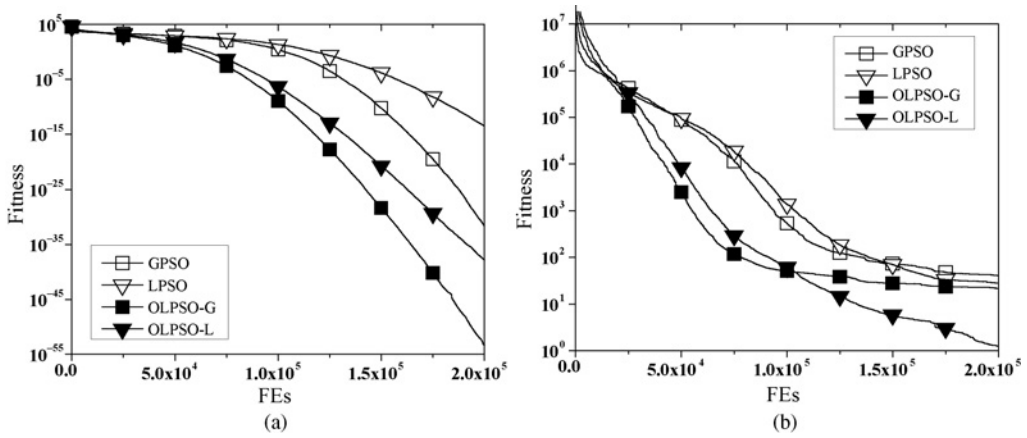

Fig. 4. Convergence progresses of the PSOs with and without the OL strategy on unimodal functions (a) f_1 (Sphere) and (b) f_3 (Rosenbrock).

optimum 0 on the Rastrigin's function (f_6) and the Griewank's function (f_8). These experimental results verify that the OLPSOs with the OL strategy offer the ability of avoiding local optima to obtain the global optimum robustly in multimodal functions.

The evolutionary progresses of the PSOs in optimizing the multimodal functions f_5 and f_6 are plotted in Fig. 5. It can be observed that OLPSOs are able to improve solutions steadily for a long period without being trapped in local optima. OLPSO-L appears to exhibit the strongest search ability and can converge to the global optimum 0 in about 1.5×10^5 FEs on the Rastrigin's function. The convergent curves on the Schewefel's function (f_5) also show that OLPSO-L has strong global search ability to avoid local optima.

3) *Rotated and Shifted Functions:* Functions f_{11} to f_{14} are multimodal functions with coordinate rotation while f_{15} and f_{16} are shifted functions. In order to avoid biases of specific rotations in the tests, a new rotation is computed before each run of the 25 independent trials. Experimental results for the four rotated multimodal functions are also given in Table V and the evolutionary progresses of f_{13} and f_{14} are plotted in Fig. 6. It appears that all the PSO algorithms are affected by the coordinate rotation. However, it is interesting to observe that the OLPSO algorithms can still reach the global optima of the rotated Ackley's function (f_{13}) and the rotated Grienwank's function (f_{14}). All the PSOs are trapped by the rotated Schwefel's function (f_{11}) and the rotated Rastrigin's function (f_{12}) as they become much more difficult after coordinate rotation [22]. However, OLPSOs still perform better than traditional PSOs on these two problems. The experimental results also show that OLPSO-G and OLPSO-L outperform GPSO and LPSO, respectively, on the two shifted function f_{15} and f_{16}. Moreover, only OLPSO-L can obtain the global optimum -330 on the shifted Rastrigin's function (f_{16}). Overall, even though affected by the rotation and the shift, the comparisons still indicate that the OL strategy is beneficial to the PSO performance, and OLPSOs generally perform better than traditional PSOs.

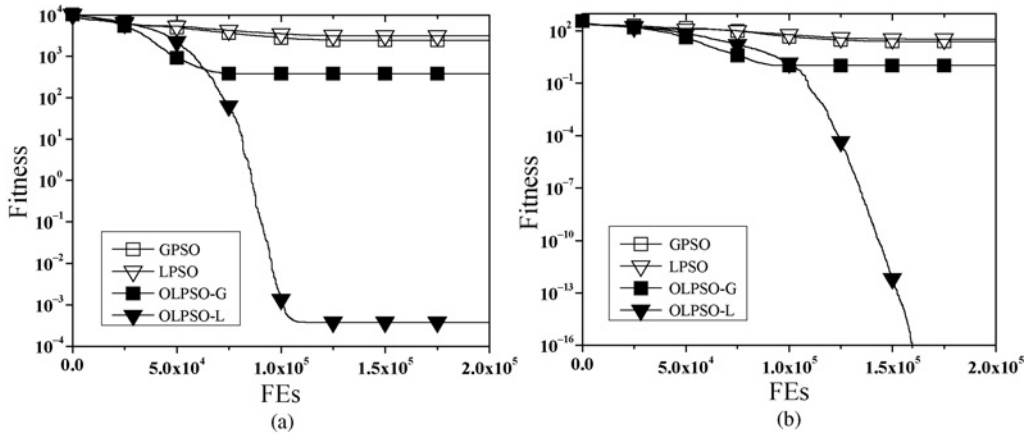

Fig. 5. Convergence progresses of the PSOs with and without the OL strategy on multimodal functions (a) f_5 (Schwefel) and (b) f_6 (Rastrigin).

C. Convergence Speed with OL Strategy

As the OL strategy can provide a promising guidance exemplar $\boldsymbol{P_o}$, it is natural that OLPSO can reach more accurate solution with a faster convergence speed. In order to verify this, more experimental results are given and compared in Table VI. The results given there are the average FEs needed to reach the threshold expressed as acceptable solutions specified in Table III. In addition, successful rate (SR%) of the 25 independent runs for each function are also compared. Note that the average FEs are calculated only for the runs that have been "successful." As some algorithms may not succeed in reaching the acceptable solution every run on some problems, the metric success performance (SP), defined as SP = (Average FEs)/(SR%) [56], is also compared in Table VI.

It can be observed from the table that OLPSO-G and OLPSO-L are constantly faster than GPSO and LPSO, respectively, on the tested functions. This indeed shows the advantage of the OL strategy in constructing promising exemplar to guide the flying direction for faster optimization speed. Moreover, with a reasonable agreement to the fact that GPSO is always faster than LPSO, OLPSO-G is observed to be faster than OLPSO-L and is also the fastest algorithm among the four contenders. Even the slower OLPSO-L (when compared with OLPSO-G), still converges faster than GPSO (global version but without OL strategy) on most of the functions. For example, in solving the Sphere function (f_1), average numbers of FEs 134 561 and 161 985 are needed by GPSO and LPSO, respectively, to reach the acceptable accuracy 1×10^{-6}. However, OLPSO-G uses only 89 247 FEs, which indicates that it is the fastest algorithm. OLPSO-L uses 98 337 FEs to obtain the solution, which is faster not only than LPSO, but also than GPSO.

The successful rates shown in the Table VI also indicate that the OL strategy is very promising in bringing a high reliability to PSO. The OLPSOs result in higher algorithm reliability with 100% successful rate on most of the test functions while traditional PSOs are sometimes trapped in the multimodal, rotated, or the shifted problems. Overall, OLPSO-L yields the highest successful rate 93.50% averaged on all the 16 functions, and followed by OLPSO-G, LPSO, and GPSO.

The experimental results have demonstrated that the OL strategy indeed can provide a much better guidance for the particles to fly to a promising region faster. The OLPSOs with the OL strategy are more robust and reliable in solving global optimization problems.

D. Comparisons with Other PSOs

In this section, the OLPSOs will be compared with some other improved PSO variants, namely, SPSO-20, SPSO-40, FIPS, HPSO-TVAC, DMS-PSO, CLPSO, and OPSO, which have been detailed in Section IV-A. The mean and the standard deviation (SD) of the final solutions are given and compared in Table VII. It can be observed that OLPSOs achieve the best solution on most of the functions. SPSO seems to be good at simple unimodal functions and SPSO-20 performs best on f_1, and f_2. Also SPSO-20 does best on the Noise function (f_4), and the shifted Rosenbrock function (f_{15}). FIPS performs best on the rotated Griewank's function (f_{14}). DMS-PSO yields the best solution on the rotated Rastrigin's function (f_{12}). CLPSO obtains the same best mean solution as OLPSO-L does on the Schewefel's function (f_5) and the shifted Rastrigin function (f_{16}). Overall, OLPSO-L performs best on f_3, f_5, f_6, f_7, f_8, f_9, f_{10}, f_{11}, f_{13}, and f_{16}, i.e., 10 out of the 16 functions.

On the unimodal functions, OLPSO-G is shown to offer superior performance among all the PSOs except SPSO. Note that the OLPSOs may not be most efficient in solving the problems with random noise, such as f_4. Similarly, OPSO which also uses an OED method (different from the OL strategy proposed in this paper) also encounters difficulties in dealing with this noisy function. This deficiency may be caused by the OED itself being fluctuated by the random noise.

On the multimodal functions, OLPSOs generally outperform all the other PSO variants. Only can OLPSO-L and CLPSO obtain high-quality mean solutions with the error value of 10^{-4} to the Schwefel's function (f_5) and the Rastrigin's function (f_6), and only OLPSO-L, CLPSO, and FIPS can obtain high-quality mean solutions with the error value of 10^{-9} to the Griewank's function (f_8).

On the coordinate rotated and shifted functions, OLPSOs also generally do better than other PSOs. OLPSO-G can still

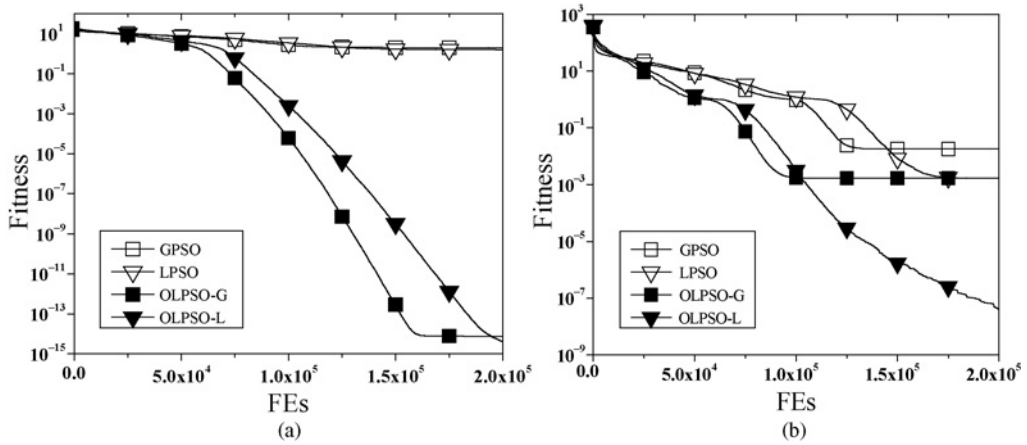

Fig. 6. Convergence progresses of the PSOs with and without the OL strategy on rotated multimodal functions (a) f_{13} (Rotated Ackley) and (b) f_{14} (Rotated Griewank).

TABLE VI
CONVERGENCE SPEED, ALGORITHM RELIABILITY, AND SUCCESS PERFORMANCE COMPARISONS

Function	GPSO			OLPSO-G			LPSO			OLPSO-L		
	FEs	SR%	SP	FEs	SR%	SP	FEs	SR%	SP	FEs	SR%	SP
f_1	134 561	100	134 561	**89 247**	100	**89 247**	161 985	100	161 985	**98 337**	100	**98 337**
f_2	141 262	100	141 262	**101 698**	100	**101 698**	171 962	100	171 962	**114 441**	100	**114 441**
f_3	126 343	100	126 343	**78 749**	100	**78 749**	137 934	100	137 934	**92 233**	100	**92 233**
f_4	171 048	**60**	**285 080**	**150 238**	40	375 595	×	0	×	**186 351**	**4**	**4 658 775**
f_5	117 710	8	1 471 375	**40 533**	**100**	**40 533**	×	0	×	**51 498**	**100**	**51 498**
f_6	75 274	100	75 274	**37 783**	100	**37 783**	76 061	100	76 061	**43 635**	100	**43 635**
f_7	152 659	100	152 659	**109 627**	100	**109 627**	189 154	100	189 154	**126 571**	100	**126 571**
f_8	137 576	32	429 925	**93 336**	**68**	**137 258.8**	171 756	80	214 695	**107 217**	**100**	**107 217**
f_9	128 474	100	128 474	**80 761**	100	**80 761**	153 943	100	153 943	**90 610**	100	**90 610**
f_{10}	135 620	88	154 113.6	**86 667**	**96**	**90 278.13**	168 060	100	168 060	**97 534**	100	**97 534**
f_{11}	77 083	76	101 425	**54 901**	**92**	**59 675**	89 029	88	101 169.3	**54 097**	**96**	**56 351.04**
f_{12}	100 215	100	100 215	**66 023**	100	**66 023**	107 072	100	107 072	**68 809**	100	**68 809**
f_{13}	163 356	16	1 020 975	**111 961**	**100**	**111 961**	×	0	×	**129 946**	**100**	**129 946**
f_{14}	146 446	32	457 643.8	**112 053**	**84**	**133 396.4**	186 771	68	274 663.2	**137 850**	**96**	**143 593.8**
f_{15}	**37 203**	84	**44 289.29**	101 632	**96**	105 866.7	**42 935**	84	**51 113.1**	113 317	**100**	113 317
f_{16}	**4758**	56	**8496.429**	37 143	**100**	37 143	**16 999**	60	**28 331.67**	43 393	**100**	43 393
Ave. SR		72.00%			**92.25%**			73.75%			**93.50%**	

Convergence speed being measured on the mean number of FEs required to reach an acceptable solution among successful runs, algorithm reliability (SR%) being the percentage of trial runs successfully reaching acceptable accuracy, and success performance (SP) is defined as the quotient of the mean FEs and SR%.

obtain the global optimum of the rotated Ackley function (f_{13}) while OLPSO-L can still obtain the global optima of the rotated Ackley (f_{13}) and the rotated Griewank (f_{14}) functions. Same as other PSOs, the OLPSO algorithms failed on the rotated Schwefel (f_{11}) and Rastrigin (f_{12}) functions, as they become much harder after rotation [22]. However, OLPSO-L is still the best algorithm on f_{11} and the results are comparable with DMS-PSO on f_{12}. Only can FIPS, DMS-PSO, OPSO, and our OLPSOs achieve the global optimum on f_{13}, only can FIPS and OLPSO-L achieve the global optimum on f_{14}, and only CLPSO and OLPSO-L achieve the global optimum on f_{16}.

Table VII also ranks the algorithms on performance in terms of the mean solution accuracy. It can be observed from the final rank that OLPSO-L offers the best overall performance, while OLPSO-G is the second best, followed by CLPSO, SPSO-40, DMS-PSO, FIPS, HPSO-TVAC, OPSO, SPSO-20, GPSO, and LPSO.

In order to compare the convergence speed, algorithm reliability, and success performance, Table VIII gives the mean FEs to reach the acceptable accuracy among the success runs, the successful rate, and the success performance. The results show that SPSO converges very fast on most of the function, while HPSO-TVAC is fastest on f_6 and f_{12}, and OPSO is fastest on f_{11}. However, OLPSOs do much better in reaching the global optima robustly, as measured by the successful rate. Even though OLPSOs are sometimes slower than SPSO and HPSO-TVAC, they are still general faster than many other PSOs. Moreover, OLPSOs generally outperform the contenders with higher successful rate. OLPSO-L has the highest successful rate of 93.50%, OLPSO-G has the second highest one of 92.25%, followed by FIPS, CLPSO, SPSO-

TABLE VII
SEARCH RESULT COMPARISONS OF PSOS ON 16 GLOBAL OPTIMIZATION FUNCTIONS

Function		GPSO	LPSO	SPSO-20	SPSO-40	FIPS	HPSO-TVAC	DMS-PSO	CLPS	OPSO	OLPSO-G	OLPSO-L
f_1	Mean	2.05×10^{-32}	3.34×10^{-14}	$\mathbf{4.08\times10^{-160}}$	2.29×10^{-96}	2.42×10^{-13}	2.83×10^{-33}	2.65×10^{-31}	1.58×10^{-12}	6.45×10^{-18}	4.12×10^{-54}	1.11×10^{-38}
	SD	3.56×10^{-32}	5.39×10^{-14}	$\mathbf{1.36\times10^{-159}}$	9.48×10^{-96}	1.73×10^{-13}	3.19×10^{-33}	6.25×10^{-31}	7.70×10^{-13}	4.64×10^{-18}	6.34×10^{-54}	1.28×10^{-38}
	Rank	6	9	**1**	2	10	5	7	11	8	3	4
f_2	Mean	1.49×10^{-21}	1.70×10^{-10}	$\mathbf{3.99\times10^{-81}}$	1.74×10^{-53}	2.76×10^{-8}	9.03×10^{-20}	1.57×10^{-18}	2.51×10^{-8}	1.26×10^{-10}	9.85×10^{-30}	7.67×10^{-22}
	SD	3.60×10^{-21}	1.39×10^{-10}	$\mathbf{1.33\times10^{-80}}$	1.58×10^{-53}	9.04×10^{-9}	9.58×10^{-20}	3.79×10^{-18}	5.84×10^{-9}	5.58×10^{-11}	1.01×10^{-29}	5.63×10^{-22}
	Rank	5	9	**1**	2	11	6	7	10	8	3	4
f_3	Mean	40.70	28.08	3.13	13.50	25.12	23.91	41.58	11.36	49.61	21.52	**1.26**
	SD	32.19	21.79	3.48	14.63	0.51	26.51	30.25	9.85	36.54	29.92	**1.40**
	Rank	9	8	2	4	7	6	10	3	11	5	**1**
f_4	Mean	9.32×10^{-3}	2.28×10^{-2}	$\mathbf{3.80\times10^{-3}}$	4.02×10^{-3}	4.24×10^{-3}	9.82×10^{-2}	1.45×10^{-2}	5.85×10^{-3}	5.50×10^{-2}	1.16×10^{-2}	1.64×10^{-2}
	SD	2.39×10^{-3}	5.60×10^{-3}	$\mathbf{1.60\times10^{-3}}$	1.66×10^{-3}	1.28×10^{-3}	3.26×10^{-2}	5.05×10^{-3}	1.11×10^{-3}	1.70×10^{-3}	4.10×10^{-3}	3.25×10^{-3}
	Rank	5	9	**1**	2	3	11	7	4	10	6	8
f_5	Mean	2.48×10^{3}	3.16×10^{3}	3.64×10^{3}	3.14×10^{3}	9.93×10^{2}	1.59×10^{3}	3.21×10^{3}	$\mathbf{3.82\times10^{-4}}$	2.93×10^{3}	3.84×10^{2}	$\mathbf{3.82\times10^{-4}}$
	SD	2.97×10^{2}	4.06×10^{2}	6.59×10^{2}	7.81×10^{2}	5.09×10^{2}	3.26×10^{2}	6.51×10^{2}	1.28×10^{-07}	5.57×10^{2}	2.17×10^{2}	**0**
	Rank	6	9	11	8	4	5	10	2	7	3	**1**
f_6	Mean	26.03	35.07	56.00	41.03	65.10	9.43	27.15	9.09×10^{-5}	6.97	1.07	**0**
	SD	7.27	6.89	15.75	11.09	13.39	3.48	6.02	1.25×10^{-4}	3.07	0.99	**0**
	Rank	6	8	10	9	11	5	7	2	4	3	**1**
f_7	Mean	1.31×10^{-14}	8.20×10^{-8}	1.28	3.73×10^{-2}	2.33×10^{-7}	7.29×10^{-14}	1.84×10^{-14}	3.66×10^{-7}	6.23×10^{-9}	7.98×10^{-15}	$\mathbf{4.14\times10^{-15}}$
	SD	2.08×10^{-15}	6.73×10^{-8}	1.00	0.19	7.19×10^{-8}	3.00×10^{-14}	4.35×10^{-15}	7.57×10^{-8}	1.87×10^{-9}	2.03×10^{-15}	**0**
	Rank	3	7	11	10	8	5	4	9	6	2	**1**
f_8	Mean	2.12×10^{-2}	1.53×10^{-3}	8.47×10^{-3}	7.48×10^{-3}	9.01×10^{-12}	9.75×10^{-3}	6.21×10^{-3}	9.02×10^{-9}	2.29×10^{-3}	4.83×10^{-3}	**0**
	SD	2.18×10^{-2}	4.32×10^{-3}	9.79×10^{-3}	1.25×10^{-2}	1.84×10^{-11}	8.33×10^{-3}	8.14×10^{-3}	8.57×10^{-9}	5.48×10^{-3}	8.63×10^{-3}	**0**
	Rank	11	4	9	8	2	10	7	3	5	6	**1**
f_9	Mean	2.23×10^{-31}	8.10×10^{-16}	0.37	7.47×10^{-2}	1.96×10^{-15}	2.71×10^{-29}	2.51×10^{-30}	6.45×10^{-14}	1.56×10^{-19}	1.59×10^{-32}	$\mathbf{1.57\times10^{-32}}$
	SD	7.07×10^{-31}	1.07×10^{-15}	0.53	3.11	1.11×10^{-15}	1.88×10^{-29}	1.02×10^{-29}	3.70×10^{-14}	1.67×10^{-19}	1.03×10^{-33}	$\mathbf{2.79\times10^{-48}}$
	Rank	3	7	11	10	8	5	4	9	6	2	**1**
f_{10}	Mean	1.32×10^{-3}	3.26×10^{-13}	0.19	1.76×10^{-3}	2.70×10^{-14}	2.79×10^{-28}	2.64×10^{-3}	1.25×10^{-12}	1.46×10^{-18}	4.39×10^{-4}	$\mathbf{1.35\times10^{-32}}$
	SD	3.64×10^{-3}	3.70×10^{-13}	0.73	4.11×10^{-3}	1.57×10^{-14}	2.18×10^{-28}	4.79×10^{-3}	9.45×10^{-13}	1.33×10^{-18}	2.20×10^{-3}	$\mathbf{5.59\times10^{-48}}$
	Rank	8	5	11	9	4	2	10	6	3	7	**1**
f_{11}	Mean	4.61×10^{3}	4.50×10^{3}	5.29×10^{3}	4.57×10^{3}	4.41×10^{3}	5.32×10^{3}	4.04×10^{3}	4.39×10^{3}	4.48×10^{3}	4.00×10^{3}	$\mathbf{3.13\times10^{3}}$
	SD	6.21×10^{2}	3.97×10^{2}	5.45×10^{2}	6.28×10^{2}	9.94×10^{2}	7.00×10^{2}	5.68×10^{2}	3.51×10^{2}	1.03×10^{3}	6.08×10^{2}	$\mathbf{1.24\times10^{3}}$
	Rank	9	7	10	8	5	11	3	4	6	2	**1**
f_{12}	Mean	60.02	53.36	56.76	43.42	1.50×10^{2}	52.90	**41.97**	87.14	63.78	46.09	53.35
	SD	15.98	13.99	19.03	17.38	14.48	12.54	**9.74**	10.76	19.73	12.88	13.35
	Rank	8	6	7	2	11	5	**1**	10	9	3	4
f_{13}	Mean	1.93	1.55	1.40	9.24×10^{-2}	3.16×10^{-7}	9.29	2.42×10^{-14}	5.91×10^{-5}	1.49×10^{-8}	7.69×10^{-15}	$\mathbf{4.28\times10^{-15}}$
	SD	0.96	0.45	1.06	0.32	1.00×10^{-7}	2.07	1.52×10^{-14}	6.46×10^{-5}	6.36×10^{-9}	1.78×10^{-15}	$\mathbf{7.11\times10^{-16}}$
	Rank	10	9	8	7	5	11	3	6	4	2	**1**
f_{14}	Mean	1.80×10^{-2}	1.68×10^{-3}	1.17×10^{-2}	3.05×10^{-3}	$\mathbf{1.28\times10^{-8}}$	9.26×10^{-3}	1.02×10^{-2}	7.96×10^{-5}	1.28×10^{-3}	1.68×10^{-3}	4.19×10^{-8}
	SD	2.41×10^{-2}	3.47×10^{-3}	1.57×10^{-2}	5.70×10^{-3}	$\mathbf{4.29\times10^{-8}}$	8.80×10^{-3}	1.24×10^{-2}	7.66×10^{-5}	3.70×10^{-3}	4.13×10^{-3}	2.06×10^{-7}
	Rank	11	5	10	7	**1**	8	9	3	4	6	2
f_{15}	Mean	427.93	432.33	**402.15**	424.28	424.83	494.20	502.51	403.07	2.45×10^{7}	424.75	415.94
	SD	54.98	43.41	**31.74**	48.94	25.37	96.54	95.18	13.50	4.40×10^{7}	34.80	23.96
	Rank	7	8	**1**	4	6	9	10	2	11	5	3
f_{16}	Mean	−223.18	−234.95	−277.47	−291.79	-245.77	−318.33	-303.17	**−330**	-284.11	−328.57	**−330**
	SD	38.58	18.82	17.31	11.18	22.08	5.75	5.01	3.39×10^{-5}	13.62	1.04	$\mathbf{1.64\times10^{-14}}$
	Rank	11	10	8	6	9	4	5	2	7	3	**1**
Ave. rank		7.38	7.50	7.00	6.13	6.56	6.75	6.50	5.38	6.81	3.81	2.19
Final rank		10	11	9	4	6	7	5	3	8	2	1
Algorithms		GPSO	LPSO	SPSO-20	SPSO-40	FIPS	HPSO-TVAC	DMS-PSO	CLPSO	OPSO	OLPSO-G	OLPSO-L

40, OPSO, DMS-PSO, LPSO, GPSO, HPSO-TVAC, and SPSO-20.

E. Comparisons With Other Evolutionary Algorithms

The proposed OLPSOs are further compared with some state of the art evolutionary algorithms (EAs) in Table IX. These algorithms include variants of EAs, such as fast evolutionary programming (FEP) with Cauchy mutation [51], orthogonal GA with quantization (OGA/Q) [42], estimation of distribution algorithm with local search (EDA/L) [57], evolution strategy with covariance matrix adaptation (CMA-ES) [58], and adaptive differential evolution (JADE) with optional external archive [59]. As OLPSO-L generally outperforms OLPSO-G in global optimization, we only compare OLPSO-L with these algorithms in Table IX. The results of the compared algorithms are all derived directly from their corresponding references except that the results of CMA-ES are obtained by our independent experiments on these functions based on the provided source code [58].

OGA/Q is an EA with orthogonal initialization and orthogonal crossover. It yielded good performance and did best on 5 of the 10 functions, indicating the advantages of the OED method. Our OLPSO-L also does best on 5 of the 10 functions. Specifically, OGA/Q seems to be better than OLPSO-L on unimodal functions (e.g., f_1 and f_2) while OLPSO-L does better than OGA/Q on some of the multimodal functions (e.g., f_9 and f_{10}). CMA-ES does best on the Rosenbrock function (f_3) and JADE did best on the Noise function (f_4). The results show that OLPSO-L has very competitive performance when compared with these state of the art EAs, especially for its strong global search ability on multimodal functions. OLPSO-L works best on f_5, f_9, and f_{10}, the same best with OGA/Q,

TABLE VIII

CONVERGENCE SPEED, ALGORITHM RELIABILITY, AND SUCCESS PERFORMANCE COMPARISONS OF PSOS ON 16 GLOBAL OPTIMIZATION FUNCTIONS

Function		GPSO	LPSO	SPSO-20	SPSO-40	FIPS	HPSO-TVAC	DMS-PSO	CLPSO	OPSO	OLPSO-G	OLPSO-L
f_1	FEs	134 561	161 985	**12 352**	20 536	118 306	63 982	138 103	139 554	92 908	89 247	98 337
	SR%	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**
	SP	134 561	161 985	**12 352**	20 536	118 306	63 982	138 103	139 554	92 908	89 247	98 337
f_2	FEs	141 262	171 962	**18 071**	29 006	165 502	78 944	147 644	172 886	134 640	101 698	114 441
	SR%	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**
	SP	141 262	171 962	**18 071**	29 006	165 502	78 944	147 644	172 886	134 640	101 698	114 441
f_3	FEs	126 343	137 934	**8277**	16 062	48 456	50 628	125 573	108 669	71 654	78 749	92 233
	SR%	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**
	SP	126 343	137 934	**8277**	16 062	48 456	50 628	125 573	108 669	71 654	78 749	92 233
f_4	FEs	171 048	×	86 978	**80 368**	91 081	×	194 220	133 550	×	150 238	186 351
	SR%	60	0	**100**	**100**	**100**	0	24	**100**	0	40	4
	SP	285 080	×	86 978	**80 368**	91 081	×	809 250	133 550	×	375 595	4 658 775
f_5	FEs	117 710	×	×	**22 640**	133 646	56 683	104 422	65 429	43 200	40 533	51 498
	SR%	8	0	0	12	**100**	92	4	**100**	4	**100**	**100**
	SP	1 471 375	×	×	188 666.7	139 214.6	61 611.96	2 610 550	65 429	1 080 000	**40 533**	51 498
f_6	FEs	75 274	76 061	6361	14 384	79 421	**6096**	74 803	44 000	24 768	37 783	43 635
	SR%	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**
	SP	75 274	76 061	6361	14 384	79 421	**6096**	74 803	44 000	24 768	37 783	43 635
f_7	FEs	152 659	189 154	**18 526**	30 290	183 341	102 496	162 400	190 767	155 088	109 627	126 571
	SR%	**100**	**100**	24	96	**100**	**100**	**100**	**100**	**100**	**100**	**100**
	SP	152 659	189 154	**77 192**	31 552.08	183 341	102 496	162 400	190 767	155 088	109 627	126 571
f_8	FEs	137 576	171 756	**12 933**	21 035	133 787	66 965	141 489	167 486	110 232	93 336	107 217
	SR%	32	80	48	64	**100**	28	56	**100**	80	68	**100**
	SP	429 925	214 695	**26 944**	32 867.19	133 787	239 160.7	252 658.9	167 486	137 790	137 258.8	107 217
f_9	FEs	128 474	153 943	17 830	**22 845**	94 368	74 033	137 909	124 779	77 587	80 761	90 610
	SR%	**100**	**100**	48	84	**100**	**100**	**100**	**100**	**100**	**100**	**100**
	SP	128 474	153 943	37 146	**27 196.43**	94 368	74 033	137 909	124 779	77 587	80 761	90 610
f_{10}	FEs	135 620	168 060	**15 390**	21 670	107 315	75 483	145 063	138 209	86 716	86 667	97 534
	SR%	88	**100**	76	84	**100**	**100**	76	**100**	**100**	96	**100**
	SP	154 113.6	168 060	**20 250**	25 797.62	107 315	75 483	190 872.4	138 209	86 716	90 278.13	97 534
f_{11}	FEs	770 83	89 029	61 346	47 672	115 196	66 394	109 220	128 544	**31 920**	54 901	54 097
	SR%	76	88	24	68	68	28	96	**100**	60	92	96
	SP	101 425	101 169.3	255 608	70 105.88	169 405.9	237 121.4	113 770.8	128 544	**53 200**	59 675	56 351.04
f_{12}	FEs	100 215	107 072	33 057	27 220	×	**8208**	88 935	146 299	107 942	66 023	68 809
	SR%	**100**	**100**	**100**	**100**	0	**100**	**100**	92	**100**	**100**	**100**
	SP	100 215	107 072	33 057	27 220	×	**8208**	88 935	159 020.7	107 942	66 023	68 809
f_{13}	FEs	163 356	×	**18 963**	30 749	187 032	×	169 314	×	161 856	111 961	129 946
	SR%	16	0	24	92	**100**	0	**100**	0	**100**	**100**	**100**
	SP	1 020 975	×	79 013	**33 422.83**	187 032	×	169 314	×	161 856	111 961	129 946
f_{14}	FEs	146 446	186 771	**17 004**	28 126	150 433	105 910	163 996	×	161 083	112 053	137 850
	SR%	32	68	36	76	**100**	32	36	0	88	84	96
	SP	457 643.8	274 663.2	**47 233**	37 007.89	150 433	330 968.8	455 544.4	×	183 048.9	133 396.4	143 593.8
f_{15}	FEs	37 203	42 935	**29 798**	44 489	75 137	129 660	140 749	129 159	75 960	101 632	113 317
	SR%	84	84	96	84	92	48	56	**100**	24	96	**100**
	SP	44 289.29	51 113.1	**31 039.58**	52 963.1	81 670.65	270 125	251 337.5	129 159	316 500	105 866.7	113 317
f_{16}	FEs	4758	16 999	**5788**	12 289	98 131	27 875	57 607	39 619	25 459	37 143	43 393
	SR%	56	60	**100**	**100**	68	**100**	**100**	**100**	**100**	**100**	**100**
	SP	8496.429	28 331.67	**5788**	12 289	144 310.3	27 875	57 607	39 619	25 459	37 143	43 393
Ave. SR		72.00%	73.75%	67.25%	85.00%	89.00%	70.50%	78.00%	87.00%	78.50%	92.25%	93.50%
SR rank		9	8	11	5	3	10	7	4	6	2	1
Algorithms		GPSO	LPSO	SPSO-20	SPSO-40	FIPS	HPSO-TVAC	DMS-PSO	CLPSO	OPSO	OLPSO-G	OLPSO-L

Convergence speed being measured on the mean number of FEs required to reach an acceptable solution among successful runs, algorithm reliability (SR%) being the percentage of trial runs successfully reaching acceptable accuracy, and success performance (SP) is defined as the quotient of the mean FEs and SR%.

TABLE IX

RESULT COMPARISONS OF OLPSO-L AND SOME STATE OF THE ART EVOLUTIONARY COMPUTATION ALGORITHMS WITH THE EXISTING RESULTS REPORTED IN THE CORRESPONDING REFERENCES

Func	FEP [51]	OGA/Q [42]	EDA/L [57]†	CMA-ES [58]‡	JADE [59]	OLPSO-L
f_1	$5.7\times10^{-4}\pm1.3\times10^{-4}$	**0±0**	N/A	$4.54\times10^{-16}\pm1.13\times10^{-16}$	$1.3\times10^{-54}\pm9.2\times10^{-54}$	$1.11\times10^{-38}\pm1.28\times10^{-38}$
f_2	$8.1\times10^{-3}\pm7.7\times10^{-4}$	**0±0**	N/A	$2.32\times10^{-3}\pm9.51\times10^{-3}$	$3.9\times10^{-22}\pm2.7\times10^{-21}$	$7.67\times10^{-22}\pm5.63\times10^{-22}$
f_3	5.06±5.87	0.75±0.11	4.324×10^{-3}	$\mathbf{2.33\times10^{-15}\pm7.73\times10^{-16}}$	0.32±1.1	1.26±1.40
f_4	$7.6\times10^{-3}\pm2.6\times10^{-3}$	$6.30\times10^{-3}\pm4.07\times10^{-4}$	N/A	$5.92\times10^{-2}\pm1.73\times10^{-2}$	$\mathbf{6.8\times10^{-4}\pm2.5\times10^{-4}}$	$1.64\times10^{-2}\pm3.25\times10^{-3}$
f_5	14.98±52.6♠	$3.03\times10^{-2}\pm6.447\times10^{-4}$♠	2.9×10^{-3}♠	$3.15\times10^{3}\pm5.79\times10^{2}$	7.1±28	$\mathbf{3.82\times10^{-4}\pm0}$
f_6	$4.6\times10^{-2}\pm1.2\times10^{-2}$	**0±0**	**0**	$1.76\times10^{2}\pm13.89$	**0±0**	**0±0**
f_7	$1.8\times10^{-2}\pm2.1\times10^{-3}$	$\mathbf{4.440\times10^{-16}\pm3.989\times10^{-17}}$	4.141×10^{-15}	12.12±9.28	$4.4\times10^{-15}\pm0$	$4.14\times10^{-15}\pm0$
f_8	$1.6\times10^{-2}\pm2.2\times10^{-2}$	**0±0**	**0±0**	$9.59\times10^{-16}\pm3.51\times10^{-16}$	$2.0\times10^{-4}\pm1.4\times10^{-3}$	**0±0**
f_9	$9.2\times10^{-6}\pm3.6\times10^{-6}$	$6.019\times10^{-6}\pm1.159\times10^{-6}$	3.654×10^{-21}	$1.63\times10^{-15}\pm4.93\times10^{-16}$	$1.6\times10^{-32}\pm5.5\times10^{-48}$	$\mathbf{1.57\times10^{-32}\pm2.79\times10^{-48}}$
f_{10}	$1.6\times10^{-4}\pm7.3\times10^{-5}$	$1.869\times10^{-4}\pm2.615\times10^{-5}$	3.485×10^{-21}	$1.71\times10^{-15}\pm3.70\times10^{-16}$	$1.4\times10^{-32}\pm1.1\times10^{-47}$	$\mathbf{1.35\times10^{-32}\pm5.59\times10^{-48}}$

†The standard deviation is not available in [57] and N/A means the results are nor available.

‡The results of CMA-ES are obtained by our independent experiments on these functions.

♠ The mean value of f_5 has been added to $418.9829\times D$ to make the global optimal value is equal to 0.

EDA/L, and JADE on f_6, the same best with OGA/Q and EDA/L on f_8, and the second best on f_7.

F. Discussions

Experimental results and comparisons verify that the OL strategy indeed helps the OLPSOs perform better than the traditional PSOs and most existing improved PSO variants on most of the test functions, in terms of solution accuracy, convergence speed, and algorithm reliability. OLPSOs offer not only better performance in global optimization, but also finer-grain search ability, owing to the OL strategy that can discover, preserve, and utilize useful information from the search experiences.

The OPSO in [48] also uses the OED method to improve the algorithm performance. The particle in OPSO uses OED on both the cognitive learning and social learning components to construct the position for the next move. The particle velocity is obtained by calculating the difference between the new position and the current position. Differently, our proposed OLPSO emphasizes the learning strategy and uses OED to design an OL strategy. The OL strategy uses OED to construct a promising and efficient exemplar to guide the particle's flying. OLPSO works under the framework of traditional PSO except that particle in OLPSO learns from its constructed guidance exemplar $\boldsymbol{P_o}$ instead of $\boldsymbol{P_i}$ and $\boldsymbol{P_n}$, i.e., uses (8) instead of (3). Therefore, the useful information in $\boldsymbol{P_i}$ and $\boldsymbol{P_n}$ can be discovered and preserved through the OL strategy.

OLPSO benefits from the following three advantages. First, since only one learning exemplar $\boldsymbol{P_o}$ is used, the guidance would be more steady and can weaken the "oscillation" phenomenon. Second, as $\boldsymbol{P_o}$ is constructed via OED on $\boldsymbol{P_i}$ and $\boldsymbol{P_n}$, the useful information can be discovered and preserved to predict promising region for guiding the particle, weakening the "two steps forward, one step back" phenomenon. Third, as the $\boldsymbol{P_o}$ is used as the learning exemplar steadily until it can not improve the particle's fitness for G generations (which has been investigated in Section III-E), it can guide the particle to fly toward the promising region steadily, resulting in better global search performance. The experimental results and comparisons support these advantages.

The comparisons between OLPSO-G and OLPSO-L show that for unimodal functions OLPSO-G outperforms OLPSO-L on both accuracy and speed, whilst for multimodal functions OLPSO-L is better for final solution accuracy. This may be due to that OLPSO-L is based on the local version PSO that provides a better diversity and avoids premature convergence. Nevertheless, OLPSO-L can also do very well on unimodal functions and it outperforms most of the existing PSOs. Hence, OLPSO-L is the recommended global optimizer here. Moreover, the comparisons with some state of the art EAs show that OLPSO-L is generally better than, or at least comparable to, these variants of EAs.

V. Conclusion

In this paper, we presented a new OLPSO by designing an OL strategy to discover useful information from a particle's personal best position $\boldsymbol{P_i}$ and its neighborhood's best position $\boldsymbol{P_n}$. This new OL strategy helps a particle construct a more promising and efficient guidance exemplar $\boldsymbol{P_o}$ to adjust its flying velocity and direction, which results in easing the "oscillation" phenomenon [17] and the "two steps forward, one step back" phenomenon [18]. OL is an operator and can be applied to PSO with any topological structures, such as the star (global version), the ring (local version), the wheel, and the von Neumann structures [15]. Without loss of generality, we applied it is to both the global and the local versions of PSO, yielding the novel OLPSO-G and OLPSO-L algorithms.

Comprehensive experimental tests have been conducted on 16 benchmarks including unimodal, multimodal, coordinate-rotated, and shifted functions. The experimental results demonstrate the high effectiveness and the high efficiency of the OL strategy and the OLPSO algorithms. The resultant OLPSO-G and OLPSO-L algorithms both significantly outperform other existing PSO algorithms on most of the functions tested, contributing to higher solution accuracy, faster convergence speed, and stronger algorithm reliability. Comparisons are also made with some state of the art EAs, and the OLPSO algorithm shows very promising performance.

The original PSO is easy to understand and implement due to its simple concept and learning strategy. The OL strategy proposed in this paper follows the same philosophy that a particle learns not only from its own experience but also from its neighbors' experiences, it is thus also easy to understand and implement.

Appendix

A. Construction of the OA with Two Levels for the D-Dimensional Problem [20], [44]

Step 1: Determine the row number $M = 2^{\lceil \log_2(D+1) \rceil}$, the column number $N = M - 1$, and the basic column number $u = \log_2(M)$.

Step 2: The elements in the basic columns are set as:

$$L[a][b] = \left(\left\lfloor \frac{a-1}{2^{u-k}} \right\rfloor \right) \bmod 2 \tag{A1}$$

where $a = 1, 2, \ldots, M$ is the row index, $b = 2^{k-1}$ is the basic column index, and $k = 1, 2, \ldots, u$.

Step 3: The elements in other columns are set as

$$L[a][b+s] = (L[a][s] + L[a][b]) \bmod 2 \tag{A2}$$

where $a = 1, 2, \ldots, M$ is the row index, $b = 2^{k-1}$ is the basic column index, $s = 1, 2, \ldots, b-1$, and $k = 2, \ldots, u$.

Step 4: For all the elements in the OA, transform the level value to 1 for the first level and the level value to 2 for the second level

$$L[a][b] = \begin{cases} 1, & \text{if } L[a][b] = 0 \\ 2, & \text{if } L[a][b] = 1 \end{cases} \tag{A3}$$

where $a = 1, 2, \ldots, M$ is the row index, and $b = 1, 2, \ldots, N$ is the column index.

References

[1] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," in *Proc. IEEE Int. Conf. Neural Netw.*, vol. 4. 1995, pp. 1942–1948.

[2] R. C. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Proc. 6th Int. Symp. Micromach. Human Sci.*, 1995, pp. 39–43.

[3] J. Kennedy, R. C. Eberhart, and Y. H. Shi, *Swarm Intelligence*. San Mateo, CA: Morgan Kaufmann, 2001.

[4] D. B. Fogel, *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence*. Piscataway, NJ: IEEE Press, 1995.

[5] Y. H. Shi and R. C. Eberhart, "Comparison between genetic algorithms and particle swarm optimization," in *Proc. 7th Int. Conf. Evol. Program.*, LNCS 1447. Mar. 1998, pp. 611–616.

[6] R. C. Eberhart and Y. H. Shi, "Particle swarm optimization: Developments, applications and resources," in *Proc. IEEE Congr. Evol. Comput.*, 2001, pp. 81–86.

[7] X. H. Hu, Y. H. Shi, and R. C. Eberhart, "Recent advances in particle swarm," in *Proc. IEEE Congr. Evol. Comput.*, 2004, pp. 90–97.

[8] Y. del Valle, G. K. Venayagamoorthy, S. Mohagheghi, J. C. Hernandez, and R. G. Harley, "Particle swarm optimization: Basic concepts, variants and applications in power system," *IEEE Trans. Evol. Comput.*, vol. 12, no. 2, pp. 171–195, Apr. 2008.

[9] M. P. Wachowiak, R. Smolikova, Y. F. Zheng, J. M. Zurada, and A. S. Elmaghraby, "An approach to multimodal biomedical image registration utilizing particle swarm optimization," *IEEE Trans. Evol. Comput.*, vol. 8, no. 3, pp. 289–301, Jun. 2004.

[10] L. Messerschmidt and A. P. Engelbrecht, "Learning to play games using a PSO-based competitive learning approach," *IEEE Trans. Evol. Comput.*, vol. 8, no. 3, pp. 280–288, Jun. 2004.

[11] N. Franken and A. P. Engelbrecht, "Particle swarm optimization approaches to coevolve strategies for the iterated prisoner's dilemma," *IEEE Trans. Evol. comput.*, vol. 9, no. 6, pp. 562–579, Dec. 2005.

[12] C. A. C. Coello, G. T. Pulido, and M. S. Lechuga, "Handling multiple objectives with particle swarm optimization," *IEEE Trans. Evol. Comput.*, vol. 8, no. 3, pp. 256–279, Jun. 2004.

[13] X. D. Li and A. P. Engelbrecht, "Particle swarm optimization: An introduction and its recent developments," in *Proc. Conf. Genet. Evol. Comput.*, 2007, pp. 3391–3414.

[14] J. Kennedy, "Small worlds and mega-minds: Effects of neighborhood topology on particle swarm performance," in *Proc. IEEE Congr. Evol. Comput.*, 1999, pp. 1931–1938.

[15] J. Kennedy and R. Mendes, "Population structure and particle swarm performance," in *Proc. IEEE Congr. Evol. Comput.*, 2002, pp. 1671–1676.

[16] J. Kennedy and R. Mendes, "Neighborhood topologies in fully informed and best-of-neighborhood particle swarm," *IEEE Trans. Syst., Man. Cybern. C*, vol. 36, no. 4, pp. 515–519, Jul. 2006.

[17] K. E. Parsopoulos and M. N. Vrahatis, "On the computation of all global minimizers through particle swarm optimization," *IEEE Trans. Evol. Comput.*, vol. 8, no. 3, pp. 211–224, Jun. 2004.

[18] F. van den Bergh and A. P. Engelbrecht, "A cooperative approach to particle swarm optimization," *IEEE Trans. Evol. Comput.*, vol. 8, no. 3, pp. 225–239, Jun. 2004.

[19] D. C. Montgomery, *Design and Analysis of Experiments*, 5th ed. New York: Wiley, 2000.

[20] Math. Stat. Res. Group, Chinese Acad. Sci., *Orthogonal Design* (in Chinese). Beijing, China: People Education Pub., 1975.

[21] Z. H. Zhan, J. Zhang, Y. Li, and H. S. H. Chung, "Adaptive particle swarm optimization," *IEEE Trans. Syst., Man, Cybern. B*, vol. 39, no. 6, pp. 1362–1381, Dec. 2009.

[22] J. J. Liang, A. K. Qin, P. N. Suganthan, and S. Baskar, "Comprehensive learning particle swarm optimizer for global optimization of multimodal functions," *IEEE Trans. Evol. Comput.*, vol. 10, no. 3, pp. 281–295, Jun. 2006.

[23] Y. H. Shi and R. C. Eberhart, "A modified particle swarm optimizer," in *Proc. IEEE World Congr. Comput. Intell.*, 1998, pp. 69–73.

[24] M. Clerc, "The swarm and the queen: Toward a deterministic and adaptive particle swarm optimization," in *Proc. IEEE Congr. Evol. Comput.*, 1999, pp. 1951–1957.

[25] M. Clerc and J. Kennedy, "The particle swarm-explosion, stability and convergence in a multidimensional complex space," *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 58–73, Feb. 2002.

[26] R. C. Eberhart and Y. H. Shi, "Comparing inertia weights and constriction factors in particle swarm optimization," in *Proc. IEEE Congr. Evol. Comput.*, 2000, pp. 84–88.

[27] P. J. Angeline, "Using selection to improve particle swarm optimization," in *Proc. IEEE Congr. Evol. Comput.*, 1998, pp. 84–89.

[28] C. F. Juang, "A hybrid of genetic algorithm and particle swarm optimization for recurrent network design," *IEEE Trans. Syst., Man, Cybern. B*, vol. 34, no. 2, pp. 997–1006, Apr. 2004.

[29] M. Lovbjerg, T. K. Rasmussen, and T. Krink, "Hybrid particle swarm optimizer with breeding and subpopulations," in *Proc. Genet. Evol. Comput. Conf.*, 2001, pp. 469–476.

[30] P. S. Andrews, "An investigation into mutation operators for particle swarm optimization," in *Proc. IEEE Congr. Evol. Comput.*, 2006, pp. 1044–1051.

[31] A. Ratnaweera, S. Halgamuge, and H. Watson, "Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients," *IEEE Trans. Evol. Comput.*, vol. 8, no. 3, pp. 240–255, Jun. 2004.

[32] R. Brits, A. P. Engelbrecht, and F. van den Bergh, "A niching particle swarm optimizer," in *Proc. 4th Asia-Pacific Conf. Simul. Evol. Learn.*, 2002, pp. 692–696.

[33] R. Brits, A. P. Engelbrecht, and F. van den Bergh, "Locating multiple optima using particle swarm optimization," *Appl. Math. Comput.*, vol. 189, no. 2, pp. 1859–1883, Jun. 2007.

[34] D. Parrott and X. D. Li, "Locating and tracking multiple dynamic optima by a particle swarm model using speciation," *IEEE Trans. Evol. Comput.*, vol. 10, no. 4, pp. 440–458, Aug. 2006.

[35] P. N. Suganthan, "Particle swarm optimizer with neighborhood operator," in *Proc. IEEE Congr. Evol. Comput.*, 1999, pp. 1958–1962.

[36] X. Hu and R. C. Eberhart, "Multiobjective optimization using dynamic neighborhood particle swarm optimization," in *Proc. IEEE Congr. Evol. Comput.*, 2002, pp. 1677–1681.

[37] J. J. Liang and P. N. Suganthan, "Dynamic multi-swarm particle swarm optimizer," in *Proc. Swarm Intell. Symp.*, Jun. 2005, pp. 124–129.

[38] T. Peram, K. Veeramachaneni, and C. K. Mohan, "Fitness-distance-ratio based particle swarm optimization," in *Proc. Swarm Intell. Symp.*, 2003, pp. 174–181.

[39] R. Mendes, J. Kennedy, and J. Neves, "The fully informed particle swarm: Simpler, maybe better," *IEEE Trans. Evol. Comput.*, vol. 8, no. 3, pp. 204–210, Jun. 2004.

[40] J. Kennedy, "Stereotyping: Improving particle swarm performance with cluster analysis," in *Proc. Congr. Evol. Comput.*, 2000, pp. 1507–1512.

[41] Q. Zhang and Y.-W. Leung, "An orthogonal genetic algorithm for multimedia multicast routing," *IEEE Trans. Evol. Comput.*, vol. 3, no. 1, pp. 53–62, Apr. 1999.

[42] Y.-W. Leung and Y. Wang, "An orthogonal genetic algorithm with quantization for global numerical optimization," *IEEE Trans. Evol. Comput.*, vol. 5, no. 1, pp. 41–53, Feb. 2001.

[43] X. M. Hu, J. Zhang, and J. H. Zhong, "An enhanced genetic algorithm with orthogonal design," in *Proc. IEEE Congr. Evol. Comput.*, 2006, pp. 3174–3181.

[44] S.-Y. Ho, L.-S. Shu, and J.-H. Chen, "Intelligent evolutionary algorithms for large parameter optimization problems," *IEEE Trans. Evol. Comput.*, vol. 8, no. 6, pp. 522–541, Dec. 2004.

[45] S.-Y. Ho, S.-J. Ho, Y.-K. Lin, and W. C.-C. Chu, "An orthogonal simulated annealing algorithm for large floorplanning problems," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 12, no. 8, pp. 874–877, Aug. 2004.

[46] S.-J. Ho, S.-Y. Ho, and L.-S. Shu, "OSA: Orthogonal simulated annealing algorithm and its application to designing mixed H2/H8 optimal controllers," *IEEE Trans. Syst., Man, Cybern. A*, vol. 34, no. 5, pp. 588–600, Sep. 2004.

[47] X. M. Hu and J. Zhang, "Orthogonal methods based ant colony search for solving continuous optimization problems," *J. Comput. Sci. Technol.*, vol. 23, no. 1, pp. 2–18, Jan. 2008.

[48] S.-Y. Ho, H.-S. Lin, W.-H. Liauh, and S.-J. Ho, "OPSO: Orthogonal particle swarm optimization and its application to task assignment problems," *IEEE Trans. Syst., Man, Cybern. A*, vol. 38, no. 2, pp. 288–298, Mar. 2008.

[49] J.-L. Liu and C.-C. Chang, "Novel orthogonal momentum-type particle swarm optimization applied to solve large parameter optimization problems," *J. Artif. Evol. Appl.*, vol. 1, pp. 1–9, Jan. 2008.

[50] S. N. Sivanandam and P. Visalakshi, "Dynamic task scheduling with load balancing using parallel orthogonal particle swarm optimization," *Int. J. Bio-Inspired Comput.*, vol. 1, no. 4, pp. 276–286, 2009.

[51] X. Yao, Y. Liu, and G. M. Lin, "Evolutionary programming made faster," *IEEE Trans. Evol. Comput.*, vol. 3, no. 2, pp. 82–102, Jul. 1999.

[52] N. Hansen and A. Ostermeier, "Completely derandomized self-adaptation in evolution strategies," *Evol. Comput.*, vol. 9, no. 2, pp. 159–195, 2001.

[53] K. Deb, A. Anand, and D. Joshi, "A computationally efficient evolutionary algorithm for real-parameter optimization," *Evol. Comput.*, vol. 10, no. 4, pp. 371–395, 2002.

[54] Y. W. Shang and Y. H. Qiu, "A note on the extended Rosenbrock function," *Evol. Comput.*, vol. 14, no. 1, pp. 119–126, 2006.

[55] *Particle Swarm Central* [Online]. Available: http://www.particleswarm.info
[56] P. N. Suganthan, N. Hansen, J. J. Liang, K. Deb, Y.-P. Chen, A. Auger, and S. Tiwari, "Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization," in *Proc. IEEE Congr. Evol. Comput*, 2005, pp. 1–50.
[57] Q. Zhang, J. Sun, E. Tsang, and J. Ford, "Hybrid estimation of distribution algorithm for global optimization," *Eng. Comput.*, vol. 21, no. 1, pp. 91–107, 2004.
[58] A. Auger and N. Hansen, "Performance evaluation of an advanced local search evolutionary algorithm," in *Proc. IEEE Congr. Evol. Comput.*, 2005, pp. 1777–1784.
[59] J. Q. Zhang and A. C. Sanderson, "JADE: Adaptive differential evolution with optional external archive," *IEEE Trans. Evol. Comput.*, vol. 13, no. 5, pp. 945–958, Oct. 2009.

Zhi-Hui Zhan (S'09) received the Bachelors degree in computer science and technology from Sun Yat-Sen University, Guangzhou, China, in 2007, where he is currently working toward the Ph.D. degree.

His current research interests include particle swarm optimization, ant colony optimization, genetic algorithms, differential evolution, and their applications in real-world problems.

Jun Zhang (M'02–SM'08) received the Ph.D. degree in electrical engineering from the City University of Hong Kong, Kowloon, Hong Kong, in 2002.

From 2003 to 2004, he was a Brain Korean 21 Post-Doctoral Fellow with the Department of Electrical Engineering and Computer Science, Korea Advanced Institute of Science and Technology, Daejeon, South Korea. Since 2004, he has been with Sun Yat-Sen University, Guangzhou, China, where he is currently a Professor and the Ph.D. Supervisor with the Department of Computer Science. He has authored seven research books and book chapters, and over 80 technical papers in his research areas. His research interests include computational intelligence, data mining, wireless sensor networks, operations research, and power electronic circuits.

Dr. Zhang is currently an Associate Editor of the IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS. He is the Chair of the IEEE Beijing (Guangzhou) Section Computational Intelligence Society Chapter.

Yun Li (S'87–M'90) received the B.S. degree in radio electronics science from Sichuan University, Chengdu, China, in 1984, the M.Eng. degree in electronic engineering from the University of Electronic Science and Technology of China (UESTC), Chengdu, in 1987, and the Ph.D. degree in computing and control engineering from the University of Strathclyde, Glasgow, U.K., in 1990.

From 1989 to 1990, he was with the U.K. National Engineering Laboratory and with Industrial Systems and Control Ltd., Glasgow. In 1991, he was a Lecturer with the University of Glasgow, Glasgow. In 2002, he was a Visiting Professor with Kumamoto University, Kumamoto, Japan. He is currently a Senior Lecturer with the University of Glasgow and a Visiting Professor with UESTC. In 1996, he independently invented the "indefinite scattering matrix" theory, which opened up a ground-breaking way for microwave feedback circuit design. From 1987 to 1991, he carried out leading work in parallel processing for recursive filtering and feedback control. In 1992, he achieved the first symbolic computing for power electronic circuit design, without needing to invert any matrix, complex-numbered or not. In 1992, he pioneered into design automation of control systems and discovery of novel systems using evolutionary learning and intelligent search techniques. He established the IEEE Computer-Aided Control System Design Evolutionary Computation Working Group and the European Network of Excellence in Evolutionary Computing Workgroup on Systems, Control, and Drives in 1998. He has supervised 15 Ph.D. students in this area and has over 140 publications.

Dr. Li is a Chartered Engineer, and a member of the Institution of Engineering and Technology.

Yu-Hui Shi (SM'98) received the Ph.D. degree in electronic engineering from South-East University, Nanjing, China, in 1992.

He is currently a Professor with the Department of Electrical and Electronic Engineering, Xi'an Jiaotong-Liverpool University, Suzhou, China. He is also the Director of the Research and Postgraduate Office, Xi'an Jiaotong-Liverpool University. Before joining Xi'an Jiaotong-Liverpool University, he was with Electronic Data Systems Corporation, Indianapolis, IN. His current research interests include the areas of computational intelligence techniques (including swarm intelligence) and their applications.

Dr. Shi is the Editor-in-Chief of the *International Journal of Swarm Intelligence Research*, and an Associate Editor of the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION. He is the Chair of the IEEE Chief Information Officer Task Force on Swarm Intelligence.